Clayson Medeiros da Silva e Matheus Duarte Dias

TÉCNICAS DE ELICITAÇÃO DE REQUISITOS: UM ESTUDO DE CASO PARA UM SISTEMA DE GESTÃO FINANCEIRA

Trabalho de Conclusão de curso submetido(a) ao Programa de Graduação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação
Orientador: Prof. Dr. Helio Aisenberg Ferenhof.

Araranguá
2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Silva, Clayson Medeiros da; Dias, Matheus Duarte
Técnicas de elicitação de requisitos : um estudo de caso para um sistema de gestão financeira / Clayson Medeiros da Silva ; Matheus Duarte Dias ; orientador, Helio Aisenberg Ferenhof, 2018.
96 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Tecnologias da Informação e Comunicação, Araranguá, 2018.

Inclui referências.

1. Tecnologias da Informação e Comunicação. 2. Elicitação de Requisitos. I. Ferenhof, Helio Aisenberg. II. Universidade Federal de Santa Catarina. Graduação em Tecnologias da Informação e Comunicação. III. Título.

Clayson Medeiros da Silva e Matheus Duarte Dias

TÉCNICAS DE ELICITAÇÃO DE REQUISITOS: UM ESTUDO DE CASO PARA UM SISTEMA DE GESTÃO FINANCEIRA

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Tecnologias da Informação e Comunicação e aprovado em sua forma final pelo Programa de Graduação da Universidade Federal de Santa Catarina

Araranguá, 19 de novembro de 2018.

Profa. Patricia Jantsch Fiuza, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Helio Aisenberg Ferenhof, Dr.
Orientador
Universidade Federal de Santa Catarina

Profª Andrea Cristina Trierweiller, Dr.ª
Universidade Federal de Santa Catarina

Prof.ª Fabiana Santos Lima, Dr.ª
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus colegas de classe e aos meus queridos pais.

AGRADECIMENTOS

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela para que hoje vislumbrássemos um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Ao professor Helio Aisenberg Ferenhof, pela orientação, apoio e confiança.

Aos nossos pais, pelo amor, incentivo e apoio incondicional, bem como a disponibilidade de tempo cedida para serem os usuários da aplicação das técnicas de elicitação de requisitos.

A todos que direta ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

RESUMO

A tarefa de elicitação de requisitos é demasiada complexa para o engenheiro de software, pois nem sempre os *stakeholders* possuem um total entendimento do domínio do problema que deve ser resolvido, ou até mesmo possuem dificuldades para demonstrar os problemas ao mesmo. Este trabalho visa elencar por meio de uma pesquisa exploratória bibliográfica nos principais autores do ramo de engenharia de software, as principais técnicas para elicitação de requisitos. Após o levantamento das técnicas, um estudo de caso é desenvolvido, onde as técnicas encontradas são aplicadas em diferentes etapas do processo de elicitação de requisitos de software. Por fim, uma análise de cada técnica de elicitação de requisitos é executada por meio do método dedutivo e indutivo, demonstrando os pontos positivos e pontos negativos bem como em qual etapa da elicitação de requisitos essas técnicas podem ser melhores empregadas. Como resultado percebeu-se que para garantir a qualidade do software, é de suma importância que a elicitação de requisitos seja tratada com atenção pela equipe de software ao início do projeto, pois deve-se evitar que novos requisitos apareçam em meio ao desenvolvimento do sistema, de forma gradual, para assim evitar custos com retrabalho e não desenvolver rotinas que não estão de acordo com a necessidade dos *stakeholders*.

Palavras-chave: Elicitação. Requisitos. Engenharia. Sistemas. TIC.

ABSTRACT

The task of requirements elicitation is too complex for the software engineer, because the stakeholders do not always have a full understanding of the problem domain that must be solved, or even have difficulties to demonstrate the problems to the problem. This work aims to list, through an exploratory bibliographical research in the main authors of the software engineering branch, the main techniques for requirements elicitation. After the survey of the techniques, a case study is developed, where the techniques found are applied at different stages of the elicitation process of software requirements. Finally, an analysis of each requirement elicitation technique is performed through the deductive and inductive method, demonstrating the positives and negatives as well as at what stage of requirements elicitation these techniques can be best employed. As a result, it was perceived that in order to guarantee the quality of the software, it is of the utmost importance that the requirements elicitation be treated with attention by the software team at the beginning of the project, since it is necessary to avoid that new requirements appear in the middle of the development of the system , in a gradual way, in order to avoid reworking costs and not to develop routines that do not meet the needs of the stakeholders.

Keywords: Elicitation. Requirements. Engineering. Systems. ICT.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Produzir um software com qualidade não significa apenas atingir os objetivos esperados pelos usuários, é uma tarefa muito mais complexa que envolve todo o ciclo de vida do sistema (BALTAZHAR, 2007).

A subjetividade de qualidade é imensa. Durante anos diversos autores definem "qualidade" de software de maneiras diferentes, Bueno e Campelo (1999) relacionam diretamente a qualidade de software às atividades de homologação presentes em todo o processo de desenvolvimento, já para Pressman (1995) a qualidade está atrelada a conformidade com os requisitos elicitados e as características implícitas de um sistema. Tavares (2006) por sua vez, afirmam que qualidade de software não significa perfeição e a classifica como um conceito multidimensional que engloba diversos atributos relacionados ao desenvolvimento, manutenção e uso. Atributos estes que devem ser pensados antes mesmo da geração do primeiro código fonte, evitando assim custos que poderiam ser reduzidos com um processo de elicitação bem estruturado.

Na visão de Bueno e Campelo (1999), os custos de qualidade podem ser divididos em 3 grandes grupos: 1) os custos de prevenção, que representam o menor custo do processo de qualidade, custos estes que vão desde o planejamento da qualidade ao treinamento; 2) os custos de avaliação que podem ser aplicados já no desenvolvimento, incluindo processos de homologação e testes, e 3) os custos de falhas, os quais são os mais críticos e com maior valor de correção, podendo ainda ter causas internas como o retrabalho ou causas externas como a resolução de queixas, suporte dentre outros.

Neste aspecto, a elicitação de requisitos tem papel fundamental no processo de desenvolvimento pois além de ser extremamente importante para guiar e auxiliar na comunicação durante a elaboração do projeto ela serve de referência para os processos de manutenção (AMBLER, 2001). Requisitos são definidos por Engholm Júnior (2010) como uma condição ou funcionalidade que deve estar presente em um software para atender determinada necessidade.

A etapa de elaboração de requisitos é uma tarefa complexa para o engenheiro de software, pois os stakeholders, ou seja, as partes interessadas, não possuem total conhecimento do domínio do problema a ser resolvido ou possuem dificuldades para expressar seus problemas ao engenheiro (PRESSMAN, 2011). Desta forma ainda segundo Pressman (2011), um dos problemas da engenharia de requisitos é a dificuldade de

entendimento da real necessidade do cliente tornando moroso e custoso o processo de elicitação de requisitos.

1.1 MOTIVAÇÃO DA PESQUISA

Na maioria das vezes as empresas de software e desenvolvedores acabam ignorando a etapa de elicitação requisitos, isso se dá pelo fato deste procedimento parecer um tanto quanto custoso. Temos como motivação para realizar essa pesquisa validar se essa teórica se comprova na prática, ou se os custos com a engenharia de requisitos podem ser considerados como um investimento, para que futuramente custos de manutenção e suporte venham a ser reduzidos.

1.2 OBJETIVOS

Com base na problemática apresentada este TCC tem por objetivo apresentar técnicas da engenharia de requisitos que auxiliem o reconhecimento das características esperadas pelos clientes e o gerenciamento do processo de validação dos requisitos com o envolvimento das partes interessadas.

1.2.1 Objetivo Geral

Analisar técnicas de engenharia de requisitos que auxiliem na elucidação e gestão dos requisitos de software

1.2.2 Objetivos Específicos

Elencar, com base na literatura, técnicas de engenharia de requisitos;

Aplicar as técnicas elencadas na elicitação de requisitos de um sistema de gerenciamento financeiro pessoal;

Testar as técnicas elencadas.

Uma vez apresentados os objetivos, passa-se a apresentar os constructos deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção se destina a apresentar o arcabouço teórico acerca dos temas seminais deste trabalho.

2.1 ENGENHARIA DE REQUISITOS

Desenvolver softwares é uma tarefa desafiadora, criativa e divertida, o que faz com que a maioria dos desenvolvedores inicie sem ter uma noção daquilo que realmente é necessário. Os argumentos utilizados pelos desenvolvedores é de que tudo ficará mais claro à medida que forem desenvolvendo o software em questão, que as coisas mudam tão rapidamente que entender os requisitos de forma detalhada é uma perda de tempo, e que todos os envolvidos no software serão capazes de entender as necessidades assim que as primeiras iterações forem desenvolvidas. (PRESSMAN, 2011).

Porém, para Pressman (2011, p. 127), "O que torna esses argumentos tentadores é que contêm elementos de verdade. Porém, cada um apresenta pontos fracos e pode levar um projeto ao fracasso."

Sendo assim, engenharia de requisitos tem como objetivo fornecer um mecanismo para que seja entendido aquilo que o cliente deseja, analisando as suas necessidades e avaliando se as necessidades são viáveis, sempre negociando uma solução que seja plausível para ambos os envolvidos no projeto, sem ambiguidades, validando as especificações e gerenciando as necessidades à medida que estas são modificadas em um sistema. (PRESSMAN, 2011).

Na visão de Hull, Jackson e Dick, (2011), a engenharia de requisitos engloba todas as partes do ciclo de vida que estão relacionadas com identificar os requisitos dos usuários do software, as análises que tem por objetivo elicitar novos requisitos, a documentação destes requisitos como uma forma de especificação e por fim, a validação destes requisitos por parte dos usuários finais do sistema a ser desenvolvido de acordo com as suas necessidades, bem como qualquer outro processo que suporta uma essas atividades apresentadas.

Para Zave (1997) conforme citado por Stiehl (1993, p. 94):

> A engenharia de requisitos é o ramo da engenharia de software preocupado com os objetivos, funções e restrições de sistemas perante o mundo real. Ela também se preocupa com o relacionamento destes fatores com especificações precisas do

> comportamento do software e com suas evoluções com o passar do tempo

Uma vez conceituada a engenharia de requisitos, é apresentada sua ligação com a engenharia de software.

2.2 ENGENHARIA DE REQUISITOS E A ENGENHARIA DE SOFTWARE

Todas as tarefas e técnicas que fazem com que os requisitos sejam conhecidos são denominados de engenharia de requisitos. Na visão do processo de desenvolvimento de sistemas, a engenharia de requisitos é uma parte da engenharia de software que tem seu início na fase de comunicação com o cliente e continua durante o processo de modelagem. A engenharia de requisitos deve ser adaptada ao processo, ao projeto, e a todos os *stakeholders*. (PRESSMAN, 2011).

Seguindo o protesto de que a engenharia de requisitos é uma atividade da engenharia de software, o relacionamento desses itens se torna fundamental para um melhor entendimento da importância da engenharia de requisitos. (PRESSMAN, 2011).

Para Sommerville (2011, p. 18):

> A engenharia de sistemas é uma atividade interdisciplinar, que envolve equipes formadas de vários antecedentes. As equipes de engenharia de sistemas são necessárias devido ao amplo conhecimento requisitado para considerar todas as implicações das decisões de projeto do sistema.

Sommerville (2011) ainda demonstra, com base na afirmação acima descrita que são necessárias várias equipes com conhecimentos multidisciplinares associados a engenharia, um exemplo dos conhecimentos necessários para o desenvolvimento de um sistema de tráfego aéreo podem ser:

- Engenharia de software;
- Engenharia eletrônica;
- Engenharia mecânica;
- Engenharia de estruturas;
- Arquitetura;
- Engenharia civil;
- Engenharia elétrica;

- Projeto de interfaces.

Para Engholm Júnior (2010) a engenharia de software se mostrou importante devido a necessidade do aumento da qualidade dos sistemas desenvolvidos e também para a diminuição dos custos e riscos envolvidos no processo de fabricação de software, bem como a criação de processos e rotinas que possam ser reaproveitadas durante todo o desenvolvimento do sistema. Pode-se afirmar que a engenharia de software veio com o objetivo de utilizar os preceitos de engenharia nos já existentes processos de desenvolvimento de software.

Ainda de acordo com Engholm Júnior (2010) devido ao aumento da complexidade e tamanho dos softwares e a necessidade da aplicação da engenharia de software, com a queda no valor dos itens de hardware, os sistemas acabaram por se tornar uma das peças mais caras no orçamento geral da computação.

Softwares de maior qualidade começaram a surgir, pois engenheiros utilizam de métodos, teorias e ferramentas para o desenvolvimento de suas atividades, das quais julgam mais apropriadas. Porém, fazem isso de forma seletiva e cautelosa, sempre buscando pela solução dos problemas mesmo quando métodos ou teorias não possam ser aplicadas. Os engenheiros sabem também que a resolução dos problemas deve respeitar certas restrições que podem ser impostas por organizações, pelo cronograma ou até por questões financeiras. (SOMMERVILLE, 2011).

Para Sommerville (2011), a engenharia de software foi concebida para ser aplicada em todas as etapas de desenvolvimento do software, desde as etapas iniciais de concepção dos requisitos até o momento em que se torna necessário realizar a manutenção do mesmo.

Devido a esse aumento de complexidade, vários problemas que antes não eram conhecidos surgiram no meio do processo de desenvolvimento de sistemas. Para Engholm Júnior (2010) dentre estes problemas, podem ser destacados:

- Imprecisão na estimativa de prazos e custos;
- Dificuldade de se atender às demandas;
- Dificuldade de se obter profissionais qualificados para atender às demandas;
- Qualidade dos softwares inferior à necessária, causando insatisfação dos usuários.

A engenharia de requisitos não tem como objetivo apenas processos técnicos do desenvolvimento de software, mas, viabiliza e

facilita os processos secundários de produção. Um exemplo de processo é a Gerência de Projetos (SOMMERVILLE, 2011).

Todas as engenharias têm como objetivo selecionar o método mais apropriado para cada atividade ou grupo de atividades, dessa forma, em algumas circunstâncias pode ser exigido uma solução mais moderada e não tão formal. Quando esse ponto não é compreendido pelas organizações e desenvolvedores, a imagem da engenharia de software acaba sendo denegrida, sendo compreendida erroneamente de processo custoso e sem retorno. (SOMMERVILLE, 2011).

Métodos alternativos e não tão técnicos às vezes podem ser solicitados, porém, uma abordagem metodizada, organizada e regularizada pode ser perfeitamente usual em vários casos, por exemplo, num grande sistema de automação industrial. Mas as mesmas técnicas utilizadas nesse sistema não se aplicariam a uma empresa com um número reduzido de funcionários que trabalha com sistemas menores, se tornando muito pesadas e custosas. (PRESSMAN, 2011).

2.3 REQUISITOS

Requisito é toda condição ou capacidade que deve ser implementada por determinado sistema ou parte deste para alcançar determinado objetivo (ENGHOLM JÚNIOR, 2010).

Para Sommerville (2011, p. 18), um exemplo de requisito para um sistema que gerencia um prédio de escritórios de forma a oferecer proteção contra incêndios e intrusos pode ser: “Fornecer um sistema de alarme contra incêndios e intrusos para o prédio que emitirá um aviso interno e externo de incêndio ou entrada não autorizada”.

Para um requisito ser válido, ele deve indicar uma necessidade direta ou indireta dos usuários e também ser aprovado diretamente pelos interessados do projeto, assegurando assim seu valor e alinhamento com os objetivos. Dessa forma, são consideradas boas práticas a documentação, organização e providenciar o acesso dos requisitos a todos os *stakeholders* no processo de desenvolvimento do sistema, garantindo que clientes e equipe de desenvolvimento estejam trabalhando com ideias alinhadas a todo o momento (ENGHOLM JÚNIOR, 2010).

Todos os projetos de software tem seus próprios requisitos de sistema, porém, um item que sempre deve ser analisado é que eles sempre devem estar alinhados com os objetivos de negócio da desenvolvedora de software, e devem ser elicitados sempre considerando estes objetivos, para garantir que o sistema tenha um rumo de negócio similar ao da empresa (ENGHOLM JÚNIOR, 2010).

Para Engholm Júnior (2010) existem alguns problemas relacionados a requisitos que podem levar o software falhas e fracassos no projeto de software, sendo eles:

- O não entendimento da real necessidade e a expectativa dos usuários pelos engenheiros ou pelos próprios usuários;
- Especificação incompleta;
- Mudança dos requisitos com o projeto em andamento ou finalizado.

Existem vários tipos de requisitos, sendo classificados entre requisitos de características ou funcionalidades esperadas pelos usuários do sistema, requisitos relacionados a desempenho, e até mesmo requisitos relacionados à segurança e confiabilidade dos dados que são manipulados no mesmo (ENGHOLM JÚNIOR, 2010).

Sommerville (2011) diz que os requisitos podem ser classificados em requisitos funcionais, requisitos não funcionais e requisitos de domínio.

2.3.1 Requisitos Funcionais

Requisitos funcionais podem ser definidos por premissas que constituem as características e as utilidades que são esperadas pelos usuários que irão utilizar o sistema. Para a demonstração e o entendimento desses requisitos, são utilizados diagramas de caso de uso A utilização destes itens normalmente envolve o atendimento a usuários e a outros sistemas (ENGHOLM JÚNIOR, 2010).

Já Sommerville (2011), destaca que os requisitos funcionais são as definições dos serviços que devem ser oferecidos pelo sistema em questão, de como ele deve reagir às entradas de dados específicas e como ele deve se comportar em determinadas situações, podendo ainda ser definido quais ações o sistema não deve tomar.

Para Pressman (2011, p. 97), é necessário um estudo de negócio, pois:

> Estabelece os requisitos funcionais e de informação que permitirão à aplicação agregar valor de negócio; define também a arquitetura básica da aplicação e identifica os requisitos de facilidade de manutenção para a aplicação.

Os requisitos irão depender do sistema que está sendo projetado, de quais serão os seus usuários e qual será a abordagem utilizada pela

equipe de desenvolvimento para elicitar os requisitos. Quando os requisitos são demonstrados como de usuário, é normal que eles apresentem uma forma mais abstrata, para que seja possível aproveitar ao máximo o compreendimento deles pelos seus interessados, porém, os requisitos também podem ser apresentados como sendo requisitos funcionais de sistema, sendo assim bem mais específicos e detalhados, demonstrando assim suas funções, entradas e saídas e até outras características (SOMMERVILLE, 2011).

Os requisitos funcionais devem descrever o que o sistema deve fazer em relação à algumas informações. Por exemplo, em uma determinada etapa de um sistema de navegação, um mapa deve ser aberto ou uma rota deve ser exibida. (SHAMIEH, 2012)

Para Sommerville (2011), os requisitos devem ser completos e consistentes. Completos no sentido de que todos os serviços solicitados pelos interessados sejam descritos, e consistentes no sentido de que os requisitos não podem ter contradições entre si. Isso se deve ao fato de que todas as características e ações do sistema terão como base as informações que foram descritas nos requisitos funcionais do projeto.

Sommerville (2011) ainda cita alguns exemplos de requisitos funcionais para um sistema, sendo eles:

- Um usuário deve ser capaz de pesquisar as listas de agendamentos para todas as clínicas;
- O sistema deve gerar a cada dia, para cada clínica, a lista dos pacientes para as consultas daquele dia;
- Cada membro da equipe que usa o sistema deve ser identificado apenas por seu número de oito dígitos.

Pressman (2011, p.253) cita que,

> Devemos estar preocupados com o desenvolvimento de uma representação de software que atenderá todos os requisitos funcionais e de desempenho e mereça aceitação com base em medidas e heurísticas de projeto.

2.3.2 Requisitos Não Funcionais

Para Sommerville (2011) requisitos não funcionais são restrições que são impostas as funções e serviços do sistema, sendo aplicados no sistema como um todo e não apenas a características ou serviços

individuais. Essas restrições incluem restrições de timing, restrições de processo de desenvolvimento, padrões, desempenho, confiabilidade, capacidade dos dispositivos de entrada e saída, entre outros.

Já para Engholm Júnior (2010), requisitos não funcionais demonstram atributos do sistema ou do ambiente do sistema, tais como:

- Capacidade de manutenção
- Reutilização de código
- Performance
- Eficiência no desenvolvimento
- Confiabilidade de dados
- Extensibilidade
- Usabilidade
- Confiabilidade
- Escalabilidade
- Reusabilidade

Os requisitos não funcionais nem sempre estão relacionados diretamente com o sistema que está sendo implementado, alguns deles estão diretamente ligados ao processo de desenvolvimento. Alguns exemplos disso são uma especificação dos padrões de qualidade, especificação de quais ferramentas de desenvolvimento devem ser utilizadas, entre outras (SOMMERVILLE, 2011).

Dentre os motivos que influenciaram o surgimento de requisitos não funcionais Sommerville (2011) destaca:

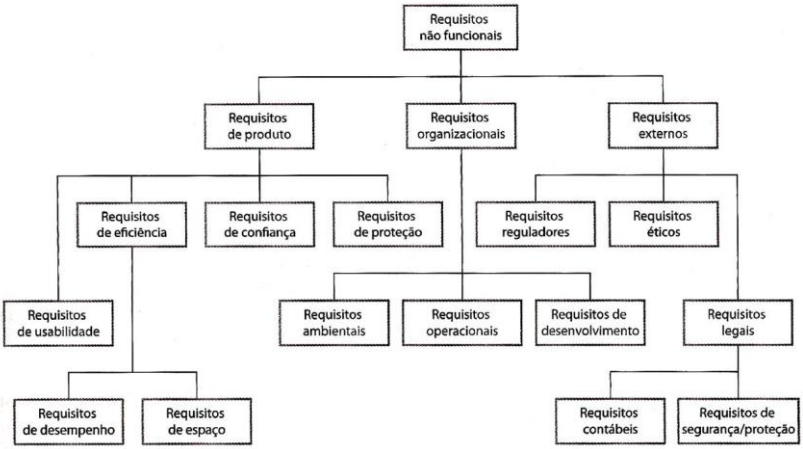
- Necessidades específicas do usuário.
- Restrições de orçamento.
- Políticas organizacionais.
- Necessidade de interoperabilidade com outros sistemas ou hardwares.
- Fatores externos, como regulamentos ou legislação.

Sommerville (2011) sugere que os requisitos não funcionais podem ser divididos em três categorias, sendo elas de requisitos de produto, quando estes se referem a características ligadas diretamente ao software, requisitos organizacionais e até requisitos externos.

- **Requisitos de produto:** Demonstram ou limitam o comportamento do sistema.
- **Requisitos organizacionais:** São ditados pelas políticas da organização dos desenvolvedores e dos clientes.

- **Requisitos externos:** São derivados de processos externos ao sistema.

Figura 1 – Tipos de requisitos não funcionais.



Fonte: Sommerville (2011).

Para Shamieh (2012), os requisitos não funcionais devem ser usados, de uma forma geral, para elicitar os requisitos de performance ou qualidade, ou até mesmo para colocar algum tipo de limite de design no software a ser desenvolvido. Dentre esses, vale destacar:

- velocidade
- capacidade
- confiabilidade
- peso
- uso
- escalabilidade

2.3.3 Requisitos de Usuário

Os requisitos de usuário devem descrever de forma não técnica, os requisitos funcionais e não funcionais de um sistema, pois esses não possuem um conhecimento técnico mais detalhado (SOMMERVILLE, 2011).

Esses requisitos devem demonstrar apenas o comportamento externo do software, sempre evitando, na medida do possível, características de projeto (SOMMERVILLE, 2011).

A principal métrica para a medição da qualidade são os requisitos de usuário. Desta forma, o descumprimento desses requisitos aponta uma falta de qualidade. (CARVALHO; TAVARES; CASTRO, 2001).

Para descrever os requisitos de usuário, devem ser utilizadas ferramentas para o fácil entendimento, com uma linguagem simples, como tabelas, formulários e diagramas intuitivos (SOMMERVILLE, 2011).

Sommerville (2011) destaca que alguns problemas podem ser encontrados ao descrever requisitos de usuário com uma linguagem muito simples, sendo eles:

- **Falta de clareza:** Pode ser difícil demonstrar os requisitos de forma clara sem torná-lo difícil de ler.
- **Confusão de requisitos:** Uma confusão pode ser gerada ao diferenciar requisitos funcionais, requisitos não funcionais, metas de sistema e informações de projeto.
- **Fusão de requisitos:** Vários requisitos podem ser expressados de forma conjunta em um único requisito.

Para que os requisitos de usuário incentivem o desenvolvedor a ter ideias inovadoras para os problemas do usuário, eles não devem ter muitas informações detalhadas, pois estas acabam restringindo a liberdade. Para isso, os requisitos de usuário devem unicamente demonstrar os recursos principais que devem ser desenvolvidos (SOMMERVILLE, 2011).

Sommerville (2011) recomenda que para evitar mal-entendidos durante a elicitação dos requisitos de usuário, algumas diretrizes simples devem ser adotadas, sendo elas:

- Utilizar um formato padrão e garantir que todas as definições de requisitos utilizem a esse formato.
- Utilizar linguagem de forma consistente.
- Utilizar destaque no texto (negrito, itálico ou cor) para ressaltar as partes principais do requisito.
- Não fazer o uso de gírias de informática.

2.3.4 Requisitos de Sistema

Requisitos de sistema são uma expansão dos requisitos de usuário, e estes são utilizados por quem está desenvolvendo o projeto como um ponto de partida para a criação do sistema. Estes requisitos são responsáveis por demonstrar de forma detalhada como os requisitos de usuário serão fornecidos pelo sistema. Eles também podem ser utilizados como forma de contrato para implementação do sistema ao usuário, sendo

assim, devem ter uma descrição completa e detalhada de todo o software. (SOMMERVILLE, 2011).

Esses requisitos definem o que o sistema deve fazer. Eles começam em um alto nível do software e são analisados e separados para que assim sejam criados os requisitos de subsistema, num nível mais baixo (SHAMIEH, 2012).

Frequentemente, uma linguagem natural é utilizada para descrever as especificações de requisitos de sistema, assim como nos requisitos de usuário. Outrora, como os requisitos de sistema são mais detalhados que os requisitos de usuário, eles podem ficar confusos utilizando essa linguagem mais natural. Sommerville (2011) cita alguns motivos pelos quais isso ocorre:

- Os conceitos das palavras utilizadas devem fazer sentido tanto para quem está desenvolvendo os requisitos quanto para os usuários.
- Uma especificação utilizando linguagem natural pode ser muito flexível e pode ter mais de uma interpretação, isso depende totalmente de o leitor definir se os requisitos estão repetidos ou distintos.
- A dificuldade de relacionar ou padronizar grupo de requisitos. Quando uma manutenção vier a ocorrer, pode ser mais fácil rever todos os requisitos do que apenas um grupo.

Devido às dificuldades apresentadas pela elicitação de requisitos de sistema com uma linguagem natural, o surgimento de mal-entendidos e conflitos se torna natural, e isso geralmente ocorre ao final do projeto, tornando a solução para o problema uma tarefa um tanto quanto cara (SOMMERVILLE, 2011).

Os requisitos de sistema podem ser demonstrados utilizando o termo “deve”, ou até mesmo utilizando de diagramas ou modelos mais elaborados (SHAMIEH, 2012).

Descrever os requisitos de uma forma natural é muito importante para o entendimento do usuário, porém, existem técnicas para a descrição desses requisitos de uma forma mais estruturada, utilizando de linguagem natural, porém, de forma organizada e estilizada, e até utilizando modelos gráficos de requisitos (casos de uso ou especificações matemáticas formais) (SOMMERVILLE, 2011).

A técnica de linguagem natural estruturada é uma maneira de escrever os requisitos onde a liberdade de quem está criando os requisitos se torna limitada, sendo todos eles escritos de uma forma padrão. Dentre as vantagens de utilizar essa linguagem, destaca-se a facilidade com que

se ocorre o entendimento da linguagem natural, porém sempre com um grau de uniformidade sendo imposto nas especificações. Utilizando linguagem estruturada, o uso de templates se torna normal e facilita na hora de limitar os requisitos, dividindo com detalhes gráficos (SOMMERVILLE, 2011).

Sommerville (2011) destaca alguns itens que devem incorporar um template de linguagem natural estruturada para elicitação de requisitos funcionais, sendo eles:

- Uma descrição da função, método, classe ou entidade que está sendo especificada.
- Descrição das entradas de dados e de onde elas vêm.
- Descrição das saídas de dados e o que vai ocorrer com elas.
- Descrição de quais outras funções ou classes são utilizadas pelo item em questão.
- Descrição de uma ação a ser tomada.
- Se está sendo descrito um requisito funcional, deve ser especificado uma pré-condição que deve ser verdadeira para que a função seja chamada, e uma pós condição, informando o que deve ser verdadeiro após a execução.
- Descrever os efeitos colaterais da operação (Se houverem).

No Quadro 1 pode-se observar um exemplo de formulário utilizando linguagem natural estruturada para descrever um requisito funcional para um sistema de bomba de insulina.

Quadro 1 – Exemplo de formulário utilizando linguagem natural estruturada.

Item	**Descrição**
Função	Calcular dose de insulina nível seguro de açúcar
Descrição	Calcula a dose de insulina a ser liberada quando o nível medido de açúcar está na zona segura entre 3 e 7 unidades
Entradas	Leitura atual do açúcar (r2), as duas leituras anteriores (r0 e r1)
Origem	Leitura atual de açúcar do sensor. Outras leituras da memória
Saídas	CompDose - a dose de insulina a ser liberada
Destino	Loop de controle principal
Ação	CompDose será zero se o nível de açúcar estiver estável ou em queda, ou se o nível estiver aumentado, mas a taxa de aumento

	estiver diminuindo. Se o nível estiver aumentando e a taxa de açúcar estiver aumentando, então CompDose será calculado dividindo-se a diferença entre o nível atual de açúcar e o nível anterior por 4, arredondando o resultado. Se o resultado do arredondamento for zero, então CompDose será definido como dose mínima que pode ser liberada.
Requer	Duas leituras anteriores de forma que a taxa de mudança do nível de açúcar possa ser calculada.
Precondição	O reservatório de insulina conter, pelo menos, o máximo da dose única permitida de insulina.
Pós-condição	r0 é substituído por r1, portanto r1 é substituído por r2.
Efeitos Col.	nenhum.

Fonte: Sommerville (2011).

2.3.5 Atributos de Requisitos

Para que a gerência dos requisitos se torne mais eficaz, é necessário que se faça a definição de alguns atributos para os requisitos, sendo a tarefa de definir quais os requisitos são necessários atribuída ao gerente de projetos e ao engenheiro de requisitos responsável (ENGHOLM JÚNIOR, 2010).

Todos os *stakeholders* devem ser responsáveis por criar e revisar cada requisito, sendo necessário que cada um tenha pelo menos os atributos que serão listados a seguir. (ENGHOLM JÚNIOR, 2010).

1. **Benefício:** Atributo responsável por indicar o grau de benefício, tendo como ponto de vista a percepção do usuário ao qual o requisito se refere.
2. **Estabilidade:** Atributo que tem como objetivo quantificar a possibilidade de mudança nele mesmo ou até em outro requisito, isso pode ocorrer devido a alteração de nível de entendimento dele pelos *stakeholders* do projeto.
3. **Situação:** Esse atributo indica em qual nível de aprovação se encontra o requisito em questão com base na opinião dos *stakeholders*.
4. **Risco:** Para esse atributo, deve ser definido quais os riscos que são oferecidos ao sistema por ele mesmo, sendo que eles serão

tratados futuramente pela equipe responsável pelo gerenciamento de riscos.

5. **Outros Atributos:** Durante o processo de desenvolvimento e da estrutura a ser utilizada para a documentação dos requisitos, é de grande importância que venham a ser utilizados outros atributos dos quais os *stakeholders* acharem necessários para que seja obtido um completo entendimento do problema em questão.

Esse padrão não precisar ser estritamente seguido, sendo que cada equipe de desenvolvimento pode definir seu próprio grupo de atributos padrão. (ENGHOLM JÚNIOR, 2010).

2.4 STAKEHOLDERS

Entende-se como *stakeholder* qualquer parte interessada em um projeto de sistema. Rozanski (2012) conceitua *stakeholder* da arquitetura de um sistema como um time, organização ou qualquer um que possua interesse na construção do sistema, partes estas preocupadas com a arquitetura, definindo essa preocupação como um requisito, um objetivo, uma restrição, uma vontade ou uma aspiração que um *stakeholder* tem para essa arquitetura.

Durante o processo de construção de um sistema, inevitavelmente várias pessoas serão afetadas pelo software e estas não estão limitadas apenas aos usuários pois um software não é apenas utilizado, ele é desenvolvido, testado, operado, muitas vezes necessita ser corrigido e melhorado, e no caminho disso há quem pague por tudo isso, dessa forma durante a vida de um sistema existe uma quantidade generosa de pessoas interessadas (ROZANSKI, 2012).

Hull, Jackson e Dick (2011), classificam o interesse dos *stakeholders* em 5 grupos: usuários, beneficiados (financeiramente ou outra vantagem), prejudicados (custos), responsáveis ou indiretamente afetados pelo uso do sistema. De forma parecida, Pressman (2011) classifica os grupos de interesse e 5, sendo eles os grupos de:

1. **Marketing:** Possui interesse nas funcionalidades, recursos e tudo que possa gerar valor no sistema.
2. **Gerentes comerciais e mantenedores:** Mais focados no negócio e nas oportunidades de ingresso no mercado, possuem interesses nos conjuntos de recursos a serem desenvolvidos de acordo com o planejamento financeiro, possuindo grande influência no produto final.

3. **Usuários finais:** Com o enfoque no produto final, os interesses dos usuários estão mais ligados à usabilidade do software e funcionalidades.
4. **Engenheiros de Software:** Se preocupam com o core da aplicação, recursos ou impeditivos técnicos não visíveis aos demais *stakeholders,* mas que possam se tornar novas funções comercializáveis.
5. **Engenheiros de Suporte:** Possuem interesse na manutenibilidade do software.

Sendo que cada uma destas ou de outras partes envolvidas contribuirá com visões e informações diferentes que devem ser coletadas, analisadas e filtradas.

Engholm Júnior (2010) separa os *stakeholders* de um projeto em dois lados, sendo eles:

- **Clientes:** Empresários, gerentes e usuários.
- **Desenvolvedores:** Gerência de projetos, estratégia de produtos (analistas de requisitos e de negócios) e fábrica de software (Arquitetos, analistas e programadores de sistemas).

E ainda classifica um terceiro grupo que possui interesse nos dois lados, sendo ele composto por equipes de controle de qualidade e equipes de implantação do sistema.

2.4.1 Fornecedores de Requisitos

Durante o processo de engenharia de requisitos um dos passos mais importantes é a identificação dos possíveis fornecedores de requisitos, ação que deve ser realizada logo no início do processo efetuando contato com setores e pessoas que poderão se tornar fontes de requisitos (ENGHOLM JÚNIOR, 2010).

Para Coutinho (2012) o processo de seleção deve começar a partir de um estudo do entorno do projeto e os seus envolvidos, estudando a documentação do que deseja ser realizado angariando dessa forma a maior quantidade possível de *stakeholders*. Engholm Júnior (2010) afirma que tais *stakeholders* devem ser de áreas e competências variadas, abrangendo desde o investidor até o usuário, passando desta forma por todas as equipes envolvidas. Engholm Júnior (2010) ainda afirma que qualquer fonte de requisitos que não for aferida poderá gerar retrabalho ou até o fracasso do projeto.

Após a fase de identificação inicial dos *stakeholders* inicia-se o processo de análise dos mesmos, levantando suas expectativas, influências e impactos no projeto (ZOPPA, 2013). Coutinho (2012) recomenda que durante o processo de gerenciamento dos *stakeholders* deve-se priorizar as partes interessadas, desenvolvendo assim estratégias para o acompanhamento dos mesmos, entendendo assim seus interesses. De forma parecida Zoppa (2013), recomenda que se identifiquem os interessados mais envolvidos com o projeto, seja por interesse direto, pela influência ou ainda por terem a capacidade de causar grande impacto no projeto.

2.4.2 Elicitação de Requisitos

O processo de elicitação de requisitos deve ser um dos primeiros passos de um projeto, possuindo um alto nível de relevância influenciando diretamente o sucesso do mesmo, e é nesta etapa que se toma conhecimento das principais expectativas dos interessados, desenvolvendo desta forma um sistema que se ajuste da melhor maneira possível aos desejos dos *stakeholders* (ENGHOLM JÚNIOR, 2010).

Para Costa e Zoucas (2012), a atividade de levantamento de requisitos é uma das mais difíceis dentro da engenharia de software, possuindo um grande impacto nas atividades subsequentes. De forma semelhante Engholm Júnior (2010), afirma que o processo de elicitação pode ser muito difícil já que se trata de uma tarefa substancialmente humana que deve levar em consideração diversos interessados e pontos de vista, podendo ainda ser prejudicada pelos próprios interessados, já que estes em sua maioria desempenham algum papel dentro de suas empresas, não possuindo assim tempo ou ainda não dando a importância que deveriam ao processo de elicitação.

O levantamento de requisitos pode ser definido como um processo de compreensão e descoberta do problema a ser resolvido pelo projeto e entendimento das necessidades de negócio a serem contempladas (ENGHOLM JÚNIOR, 2010). Por sua vez Moraes (2010), afirma que a elicitação de requisitos é o processo de busca, descobrimento e obtenção de requisitos para sistemas, e ainda afirmam que requisitos não incluem apenas necessidades dos usuários, mas também as necessidades que surgem de padrões organizacionais, governamentais ou industriais em geral.

Engholm Júnior (2010) lista algumas técnicas que podem ser utilizadas para o processo de obtenção das informações necessárias para produção de requisitos:

- Entrevistas
- Reuniões
- Análise de documentos
- Análise de sistemas utilizados anteriormente
- Observação das tarefas executadas
- Brainstorming
- Dentre outras.

Engholm Júnior (2010) ainda recomenda que seja efetuada a formalização de todas as informações obtidas em documentos, possibilitando assim a concordância entre todas as partes envolvidas no processo de criação do sistema.

Após a análise inicial dos dados e informações obtidas deve-se realizar a filtragem e o alinhamento das regras de negócio, eliminando assim requisitos duplicados ou sem embasamento, sendo que todos os procedimentos adotados durante a elicitação devem ser registrados um plano de gerenciamento de requisitos, informando nele todas as orientações necessárias para entendimento e uso dos requisitos, além dos procedimentos a serem utilizados no processo de elicitação, rastreamento dentre outros (ENGHOLM JÚNIOR, 2010).

2.4.2.1 Elicitação em Alto Nível

A documentação obtida a partir dos primeiros contatos entre times de desenvolvimento e demais interessados tem por nome Elicitação em Alto nível, sendo ela a responsável por delimitar e levantar as peculiaridades, bem como entender as expectativas a serem cumpridas (ENGHOLM JÚNIOR, 2010).

De acordo com Engholm Júnior (2010) a elicitação de alto nível pode ser dividida em 4 atividades:

- **Compreensão do problema:** Uma das principais etapas de um projeto pois o resultado dela irá guiar todo o processo de desenvolvimento.
- **Identificação dos *stakeholders*:** Nesta etapa é crucial que todos os interessados sejam identificados e examinados.
- **Levantamento das expectativas:** Após a compreensão do problema e análise dos interessados deve-se levantar as expectativas para com o produto final.
- **Criação do glossário do projeto:** A comunicação é o ponto chave para o sucesso do projeto, desta forma, definir um vocabulário comum entre as equipes com abreviações, siglas e

demais termos técnicos é de extrema importância para o bom andamento do projeto.

3 MÉTODOLOGIA DE PESQUISA

Na ciência, metodologias científicas são instrumentos básicos que conduzem de forma ordenada a maneira que deve proceder um cientista para que alcance um determinado objetivo ao longo de um percurso. Pode ser considerada um traço característico da ciência aplicada (FACHIN, 2005).

Sem a utilização de um método científico ficaria impossível falar de ciência, pois não seria possível colocar em evidência os processos operacionais utilizados para que o objetivo científico fosse atingido (FACHIN, 2005).

Com o intuito de atingir o objetivo em apresentar técnicas de engenharia de requisitos que auxiliem na elucidação e gestão dos requisitos de software, este trabalho de conclusão de curso, tomou base na revisão narrativa, também denominada revisão exploratória da literatura. De acordo com Ferenhof e Fernandes (2016), este tipo de revisão de literatura, não necessita da definição de critérios explícitos para busca de documentos. A seleção dos mesmos não segue uma sistemática e, é feita de forma arbitrária. O pesquisador pode incluir documentos de acordo como seu viés, além disto, não há preocupação em esgotar as fontes de informação.

Para analisar as técnicas de elicitação de requisitos encontradas por meio da busca exploratória na literatura, foram utilizados os métodos indutivo e dedutivo, em um grupo de foco.

Destaca-se que o método do grupo de foco é uma entrevista com várias pessoas sobre um tópico ou assunto específico (BRYMAN, 2016).

O grupo foco selecionado para a realização da aplicação prática foram seis potenciais usuários totalmente leigos no assunto de desenvolvimento de software. Dos selecionados apenas dois com formação em ensino superior e quatro com formação em ensino médio. A faixa etária dos selecionados ficou entre quarenta e cinquenta anos.

Isso se deu para que a aplicação prática do trabalho acontecesse na forma mais próxima de um ambiente de relacionamento entre empresa de software/cliente possível. O tempo de aplicação das técnicas demorou demasiado diferente para cada uma, sendo que algumas técnicas não necessitavam do ajuntamento dos *stakeholders* fisicamente. A aplicação das técnicas ocorreu no segundo semestre de 2018.

O método dedutivo é um método racional. Métodos racionais são todos que fazem parte da estrutura do raciocínio, ou seja, que coletam elementos relativos que podem ser analisados pela faculdade espiritual do homem, que é a razão (FACHIN, 2005).

O método indutivo por sua vez é um processo que parte dos dados para temas mais abrangentes, onde o pesquisador reúne informações detalhadas separando-as em categorias ou temas que serão desenvolvidos em padrões amplos, teorias ou generalizações as quais são comparadas com as experiências pessoais ou com a literatura do tema (CRESWELL, 2007).

4 TÉCNICAS PARA ELICITAÇÃO DE REQUISITOS

Este capítulo apresenta o resultado da revisão exploratória da literatura, explicitando técnicas para levantamento de requisitos.

4.1 ENTREVISTAS

Entrevistar *stakeholders*, seja de maneira formal ou informal é uma das atividades mais realizadas durante o processo de engenharia de requisitos e são nas entrevistas que a equipe de desenvolvimento consegue estabelecer um comparativo entre o atual quadro e a projeção de sistema que se deseja alçar (SOMMERVILLE, 2011).

Para Kendall e Kendall (1992) as entrevistas são importantes pois conseguem abranger uma grande quantidade de interessados, obtendo assim dados necessários para correções e melhorias no sistema a partir de diferentes pontos de vista.

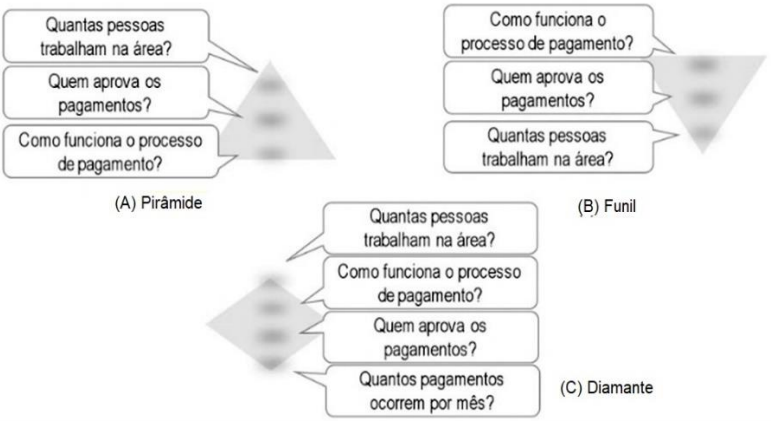
Sommerville (2011) divide as entrevistas em dois tipos que podem ser aplicadas de maneira conjunta, sendo elas:

- **Fechadas:** Entrevistas com questões predefinidas.
- **Abertas:** As perguntas não são predefinidas permitindo assim a equipe de desenvolvimento explorar de várias formas para obter o entendimento das necessidades a serem cumpridas.

De forma a complementar este pensamento, Vazquez e Siqueira (2016) definem 3 maneiras básicas de estruturar uma entrevista sendo elas:

A. **Pirâmide:** Este método inicia com questões fechadas e novas questões abertas vão sendo inseridas à medida que a entrevista progride, sendo esta uma das formas de "quebrar o gelo" entre o entrevistador e o entrevistado.
B. **Funil:** Ao contrário do método da pirâmide, este inicia-se com questões abertas que vão fechando ao decorrer da entrevista e deve ser utilizada quando o entrevistado deseja ir direto ao ponto.
C. **Diamante:** Combinando as duas técnicas anteriores o método diamante inicia com questões fechadas, abrindo-as ao decorrer e finalizando com novas questões fechadas, sendo em geral a melhor forma de estruturar uma entrevista, entretanto é também, o método mais longo.

Figura 2 – Maneiras básicas de estruturar uma entrevista.



Fonte: Vazquez e Siqueira (2016).

Martins (2001), coloca que por serem mais flexíveis, as entrevistas conseguem extrair de melhor forma as informações de caráter subjetivo, além disso aproximam os usuários e desenvolvedores. Em contrapartida, o custo e tempo investido podem ser prejudiciais caso haja um grande número de usuários, causando inclusive uma dificuldade de organização dos dados obtidos, já para Sommerville (2011) as entrevistas são úteis para compreensão geral das ações executadas pelos envolvidos bem como os mesmos desejam interagir, elencando problemas e soluções de softwares anteriores, porém não são tão úteis quando se trata de requisitos do domínio da aplicação.

O processo de elicitação do conhecimento do domínio pode ser dificultado por duas razões segundo Sommerville (2011):

- *Stakeholders* podem julgar determinado conhecimento de domínio óbvio por já estarem habituados a ele e acabam não descrevendo características do mesmo.
- As terminologias utilizadas por especialistas podem dificultar o entendimento por parte dos engenheiros de requisitos.

O levantamento de requisitos por meio de entrevistas é um processo pobre por não atingir algumas informações essenciais e devem ser utilizados de forma suplementar a outras técnicas de recolhimento de informações as quais a equipe de engenharia de requisitos julgar adequadas ao software a ser desenvolvido (SOMMERVILLE, 2011).

4.2 ETNOGRAFIA

A Etnografia é o ato do analista de requisitos se inserir no ambiente o qual o sistema a ser desenvolvido será utilizada para assim observar e documentar processos operacionais ajudando a extrair requisitos de apoio para tais processos (SOMMERVILLE, 2011).

Para Fabri (2012, p. 1) a etnografia é vista como “uma técnica de observação utilizada para mapear requisitos implícitos que refletem processos reais dentro de um ambiente sistêmico", de maneira semelhante Sommerville (2011) aponta que softwares estão inseridos dentro de um contexto social e organizacional, desta forma os requisitos deste sistema podem ser derivados ou restringidos desse contexto já que cumprir estes requisitos é importante para o sucesso do projeto.

A Etnografia se destaca das demais técnicas de levantamento de requisito pois ajuda a levantar requisitos implícitos e a entender na prática como determinados processos são utilizados já que o uso real do sistema pode divergir parcial ou completamente do processo desenhado (SOMMERVILLE, 2011). De forma complementar Vazquez e Siqueira (2016) colocam que ao inserir o analista de requisitos na vivência do problema fica mais fácil compreender as dificuldades enfrentadas, a importância e a criticidade do trabalho realizado.

Entretanto, o uso de observações etnográficas é limitado a processos já existentes que possuam algo a ser observado, além disso, a etnografia é uma tarefa muito onerosa já que demanda de muito tempo. Outro ponto importante é que processos que ocorram com baixa frequência podem não acontecer nas janelas de observação, desta forma para analisar de forma completa é necessário um tempo aquém do disponível nos projetos de software (VAZQUEZ; SIQUEIRA, 2016).

Utilizar de estudos etnográficos é relevante para o processo de levantamento de requisitos pois pode revelar detalhes cruciais de processos operacionais que podem não serem obtidos em outras técnicas de elicitação, todavia, devido ao seu foco no uso final do sistema esta abordagem não é adequada para levantar requisitos de domínio da aplicação. Isto deve-se ao fato de nem sempre a inserção no ambiente de trabalho conseguir identificar novos recursos que podem ser inseridos no sistema, tornando-a incompleta, devendo então ser utilizada em complemento a outras técnicas (SOMMERVILLE, 2011).

4.3 CENÁRIOS

Os usuários de sistemas possuem maior facilidade para compreender exemplos reais a descrições abstratas, desta forma, a utilização de cenários operacionais, os quais demonstram a forma como as pessoas podem interagir com o software aumentam a capacidade de compreender, avaliar e criticar as características do mesmo (SOMMERVILLE, 2011).

Para Vazquez e Siqueira (2016, p. 241) um cenário “corresponde a diferentes passos que se desdobram a partir de um evento que inicia o caso de uso e das condições que afetam como ele deve se comportar.". De forma semelhante Silva e Videira (2001), definem cenários como uma instância de um caso de uso, na qual um caso de uso pode possuir dezenas de cenários.

A medida que o projeto avança e os requisitos são elicitados começa-se a construir uma visão geral do sistema, porém, para que a equipe de desenvolvimento consiga progredir com qualidade é necessário que haja o entendimento das funções e características do sistema e como elas serão utilizadas pelos diferentes tipos de usuários. Desta forma a criação de cenários que identifiquem os possíveis roteiros de uso se torna importante (SOMMERVILLE, 2011).

Para Stiehl (2011), cenários devem ser utilizados para adicionar detalhes a uma descrição geral dos requisitos já elicitados, devendo cobrir apenas uma parte das interações. Desta forma torna-se necessária a criação de vários cenários visando oferecer uma gama de informações sob diferentes níveis de detalhamento.

Sommerville (2011) ainda afirma que um cenário é iniciado por meio do seu esboço de interação, e ao longo do processo de levantamento de requisitos mais detalhes são adicionados a este esboço, desta forma obtém-se uma descrição completa da interação, a qual pode ser exposta de forma geral em cinco descrições esperadas pelo usuário:

- O que esperar quando o sistema iniciar;
- Fluxo normal dos eventos;
- Falhas e tratamentos;
- Processos que podem acontecer simultaneamente;
- Estado do sistema ao término do processo.

De forma semelhante Vazquez e Siqueira (2016) afirmam que a descrição de um cenário não deve abordar detalhes como interface gráfica, hardware, software ou requisito não funcional, mas deve explorar quatro pontos básicos:

- Quando e como inicia-se um cenário;
- Quais as possíveis interações entre atores e o cenário;
- Quando se fará o uso de referências ou manutenção de dados;
- Quando e como o cenário se encerra.

O processo de levantamento de requisitos embasado em cenários tem como principal característica a participação dos *stakeholders* para identificação de cenários e detalhamentos. Usualmente tais cenários podem ser descritos de forma textual, todavia há também uma abordagem mais completa e estruturada a qual pode utilizar os casos de uso (STIEHL, 2011).

4.4 CASOS DE USO

Caso de uso é definido por Silva e Videira (2001, p. 146) como uma sequência de ações realizadas por um ou mais atores de modo a obterem um resultado desta forma os mesmos afirmam que:

> O modelo de casos de utilização permite capturar os requisitos de um sistema por meio do detalhe de todos os cenários que os utilizadores podem realizar. Os casos de utilização, mais que iniciar a modelação de requisitos de um sistema, dirigem/conduzem todo o processo de desenvolvimento.

Sommerville (2011) descreve casos de uso como uma interação entre o sistema e um ou mais agentes externos, conhecidos como atores. Esta interação pode receber algumas informações adicionais, podendo ser informações textuais ou utilizar modelos gráficos. Além disso o Caso de Uso é uma técnica para levantamento de requisitos que apesar de hoje possuir forte ligação com a notação UML, não foi criada por ela e sim adaptada.

Para Fowler (2005) o verdadeiro sentido do caso de uso é a sua descrição textual utilizada na forma de cenários, fugindo assim dos conceitos apresentados no UML. A notação foca na apresentação do diagrama de caso de uso, e os casos relacionados, mas não descreve a maneira como devem ser capturados os conteúdos, trazendo assim informações limitadas para equipe de desenvolvimento.

Existem duas formas básicas de especificar casos de uso, onde uma delas considera cada caso de uso como um cenário único e a outra prega

que um caso de uso pode conter mais de um cenário criando assim segmentos com um caso principal e demais cenários alternativos e é na fase de elicitação de requisitos que todos os casos de uso de alto nível são arquivados em um único documento, representando assim todas as possibilidades de interações do sistema (SOMMERVILLE, 2011).

Para fornecer uma análise completa das características impactadas, Sommerville (2011) recomenda que todos os casos de uso levantados sejam documentados com uma descrição textual completa, permitindo desta forma compreender todo o funcionamento das interações e seus impactos, podendo ainda referenciar outros documentos ou artefatos para uma completa documentação das características.

4.5 BRAINSTORMING

Pode-se dizer que entre as técnicas de reunião em grupo para elicitação de requisitos, o Brainstorming é uma das mais conhecidas. Essa técnica consiste em realizar uma reunião entre vários especialistas de diversificados setores, de forma que cada um tenha a função de estimular no próximo a criação de ideias, sempre tendo em vista solucionar o problema que está sendo colocado em discussão. Sob nenhuma hipótese alguma ideia deve ser criticada. Geralmente o Brainstorming é utilizado nas fases iniciais do projeto, pois essa etapa sempre é a mais carente de ideias, sendo vital o levantamento de opiniões acerca de diversos assuntos (BATISTA; CARVALHO, 2003).

O brainstorming tem como objetivo gerar novas ideias por meio da liberdade de criação e aceitação entre os envolvidos na reunião, sendo que uma reunião bem-sucedida de brainstorming tem como resultado um bom acervo de novas ideias, sendo que os participantes sentem que cada um, de alguma maneira, contribuiu para a solução definitiva do problema em questão. Um brainstorming possui uma alta eficácia quando se trata de uma concepção de um sistema (BATISTA; CARVALHO, 2003).

Essa técnica de chuva de ideias contém duas fases, sendo uma a fase de geração de ideias, onde as ideais são coletadas, e a fase de avaliação, onde as ideias coletadas são discutidas entre os *stakeholders*. Essa separação em duas fases se torna importante para que a primeira fase, que é a de geração, não tenha nenhum tipo de avaliação ou críticas, pois mesmo uma ideia não sendo boa, essa pode servir de base para criação de novas ideias, contribuindo sempre para a solução do problema comum (PETRY; GARCIA; SOUZA, 2014).

Em contrapartida Goguen (1997) coloca que a diferença da hierarquia ou de interesses pessoais dos participantes podem atrapalhar,

fazendo com que os participantes não se sintam à vontade para expor determinados conhecimentos.

Os casos de uso possuem um alto custo de esforço pois necessita que um grupo significativo de interessados esteja reunido, além disso as reuniões de brainstorm são extremamente sensíveis e podem ser comprometidas facilmente caso aconteça algum tipo de crítica às ideias apresentadas (BATISTA; CARVALHO, 2003). Outro fator negativo é que nem sempre os participantes contribuirão com todo o seu conhecimento sobre o assunto, pois os mesmos podem se sentir intimidados com os demais presentes na reunião (GOGUEN, 1997).

Contudo pode ter seu aproveitamento expandido caso seja utilizado nas fases iniciais de um projeto ou módulo e com usuários com alguma experiência em brainstorm (BATISTA; CARVALHO, 2003).

5 VERIFICAÇÃO PRÁTICA DAS TÉCNICAS DE ESPECIFICAÇÃO DE REQUISITOS

Afim de verificar a aplicabilidade prática das técnicas de especificação de requisitos, este trabalho buscou especificar os requisitos necessários para o desenvolvimento de um aplicativo/*software* denominado OrganiApp - Gestão Financeira Pessoal.

Para tal, foram utilizadas de cinco técnicas de elicitação de requisitos levantadas por meio de uma pesquisa exploratória bibliográfica e apresentadas na seção quatro, no grupo focal descrito na seção três.

A escolha das etapas para realizar a aplicação de cada técnica se deu baseando-se nas indicações dos autores, conforme revisão bibliográfica.

O objetivo desta seção, é demonstrar como ocorreu a aplicação de cada técnica de elicitação, bem como demonstrar vantagens e desvantagens e qual a utilidade indicada de cada técnica, comparando com as indicações da literatura, por meio do método indutivo e dedutivo.

Um aplicativo de gerenciamento financeiro pessoal foi escolhido devido a uma pesquisa realizada com o grupo foco, onde a maioria optou por um software deste tipo devido a carência do mesmo no mercado e também porque tinham dificuldade de gerenciar suas despesas e receitas, utilizando de métodos arcaicos que poderiam ser substituídos por soluções digitais, no caso o OrganiApp.

Toda documentação gerada a partir da aplicação das técnicas de elicitação de requisitos está anexado em forma de apêndices.

5.1 ENTREVISTAS

Na visão geral, juntamente com o *brainstorm* utilizou-se a técnica de entrevistas. O modelo de entrevistas utilizado foi o de *Funil*, modelo este sugerido por Vazquez e Siqueira (2016). Essa técnica foi utilizada para definir um conceito geral do sistema a ser desenvolvido, para assim posteriormente a ideia ser incrementada por meio do *brainstorm.*

Inicialmente perguntas abertas foram feitas, buscando entender o problema do usuário com a organização das suas finanças de uma forma mais geral e abstrata. Posteriormente, perguntas de sentido mais direcionado foram feitas, como qual o número de lançamentos que seriam realizados por competência (necessário para definir a organização do dashboard), necessidade de relatórios e quais tecnologias poderiam ser utilizadas para o desenvolvimento do sistema.

Os documentos com os questionários das entrevistas encontram-se anexados no Apêndice C.

Para a elicitação dos requisitos do Dashboard, também foram utilizadas entrevistas com os usuários, porém desta vez com uma abrangência maior de entrevistados, indo de encontro a Kendall e Kendall (1992) onde apresenta que as entrevistas são importantes pois conseguem abranger uma grande quantidade de interessados, e levando em conta a importância do dashboard, uma escolha entre sugestões de uma forma mais democrática e de diferentes pontos de vista foi interessante.

Ao utilizar as entrevistas pode-se perceber como vantagem uma maior interação dos usuários e a equipe de desenvolvimento por quebrar a barreira entre stakeholders e desenvolvedores (MARTINS, 2001). Outro ponto forte do uso de entrevista é que abrangem uma grande quantidade de interessados de uma vez só (KENDALL; KENDALL, 1992) desta forma facilitam a compreensão geral do sistema e de como os usuários/interessados pretendem interagir com o sistema (SOMMERVILLE, 2011). Além disso, entrevistas conseguiram extrair com mais eficiência informações subjetivas a respeito do sistema (MARTINS, 2001).

As entrevistas de acordo com Sommerville (2011) são úteis para a compreensão geral das ações a serem executadas pelos usuários e até mesmo para entender como os mesmos esperam agir no sistema, obtendo dessa maneira quais as informações que deveriam ser priorizadas no *dashboard.*

Para obtenção dos dados foram utilizadas as duas formas de entrevistas propostas por Sommerville (2011), com questões abertas e fechadas, baseado nisso também, utilizou-se da estrutura Diamante proposta por Vazquez e Siqueira (2016) a qual propõe a inicialização da entrevista com questões fechadas, partindo para questões abertas e finalizando com novos questionários fechados.

A utilização da estrutura de Diamante se mostrou muito eficiente pois ao iniciar com questões fechadas existe uma quebra da resistência do entrevistado, permitindo que ele tomasse a liberdade para expor pontos de vista diferentes ao longo da entrevista contudo limitando-o novamente ao fim, porém o tempo utilizado por entrevista foi um fator negativo, conforme já previsto por Vazquez e Siqueira (2016).

A partir da aplicação da técnica de entrevista, foi possível perceber na prática que quando utilizada de maneira isolada, ela não atinge algumas informações essenciais para o levantamento dos requisitos, porém quando utilizada de forma conjunta com outras técnicas, como foi no caso da “visão geral”, onde foi mesclada com o brainstorm, se mostrou

muito mais eficiente, levantando um maior número de requisitos. Como resultado dessa análise, formou-se o Quadro 7.

5.2 ETNOGRAFIA

A técnica de Etnografia foi utilizada para elicitar os requisitos da etapa de relatórios (vide Apêndice A), pois o foco desta ferramenta é se inserir no ambiente do usuário para entender melhor quais são suas dificuldades e assim tentar perceber quais resultados ele espera do sistema ao fim dos cadastros e processos. Segundo Sommerville (2011) a inserção do analista de requisitos na vivência do problema facilita a compreensão das dificuldades encontradas pelos usuários.

A utilização desta técnica se deu pelo fato de que a maioria das vezes os requisitos divergirem parcial ou até completamente do que foi desenhado, fazendo com que os resultados demonstrados pelos relatórios não sejam satisfatórios para a real utilização do usuário (SOMMERVILLE, 2011).

Para aplicação desta técnica, selecionou-se um grupo de pessoas, e foi realizado um acompanhamento de como os usuários geriam seus recursos financeiros, onde esses recursos eram anotados, se eram por meio de planilhas ou então de anotações em cadernos, e quais os resultados eram esperados por eles após a organização dos seus lançamentos. Ao fim de um determinado período os usuários reservaram um tempo para colocar suas planilhas e anotações em dia, nesse período, foi solicitado que aos usuários que chamassem os responsáveis pela elicitação, para que o acompanhamento fosse realizado.

A partir da vivência e percepção de como os usuários geriam seus recursos financeiros obteve-se então um apanhado de informações relevantes para definição do funcionamento dos filtros a serem disponibilizados e das informações a serem retornadas.

Por ter foco no usuário final, conforme Sommerville (2011) a utilização da etnografia facilitou o entendimento da real necessidade dos usuários, porém é um processo moroso, pois além da observação existe a etapa de análise e documentação das informações.

A inserção no ambiente permite uma melhor percepção dos problemas, porém bloqueia a criatividade quando se trata da sugestão de novas maneiras de resolver o problema do usuário, tornando a técnica de etnografia incompleta. Essa ferramenta seria melhor aproveitada se fosse mesclada com alguma outra técnica de elicitação, desta forma esta outra técnica poderia incrementar algumas novas maneiras de gerenciar os dados cadastrados no sistema, não apenas digitalizando os processos que

o usuário já utilizava manualmente. Essa análise serviu para construção do Quadro 3.

A documentação recolhida após a aplicação da etnografia encontra-se no Apêndice F.

5.3 CENÁRIOS

Para dar forma a tela de lançamentos, que inicialmente foi formulada com uma visão geral (vide Apêndice A), por meio de brainstorm e entrevistas, foi utilizada a técnica de cenários juntamente com casos de uso, pois conforme Stiehl (2011) os cenários devem ser utilizados para dar detalhes a uma descrição geral dos requisitos já levantados.

Para criação dos cenários não foram abordados, conforme Vazquez e Siqueira (2016), hardware, software ou requisitos não funcionais. Para definir os cenários e suas características utilizaram-se os itens definidos por Sommerville (2011) que são: O que esperar quando o sistema iniciar (pré-condição), fluxo normal dos eventos, falhas e tratamentos, processos que podem acontecer simultaneamente e estado do sistema ao término do processo (pós-condição).

Juntamente com os cenários, foram desenvolvidos casos de uso para facilitar a identificação dos atores e até mesmo dos cenários, pois por meio dos casos de uso foram criados uma espécie de esboço para assim então a criação dos cenários, que foram uma implementação dos casos de uso. Esse método de utilização foi sugerido por Fowler (2005) que diz que o verdadeiro sentido do caso de uso é ter sua descrição textual utilizada na forma de cenários, fugindo assim dos conceitos apresentados no UML.

Com a utilização dos cenários, pode-se perceber que eles se davam como estórias que explicavam como o sistema deveria ser utilizado, dando entendimento aos stakeholders de uma forma mais concreta e usável como o processo deveria ocorrer, bem como quais seriam as ações dos usuários e também do sistema perante algumas situações descritas no mesmo.

A dificuldade encontrada foi descrever os fluxos dos eventos tendo em vista que anteriormente os usuários selecionados para a participação utilizavam-se de planilhas e anotações para gerenciar suas despesas e receitas, dessa forma notou-se uma maior participação dos técnicos para sugerir fluxos de trabalho visando a facilitação e categorização dos lançamentos e posteriormente uma aprovação ou rejeição dos participantes.

Fica difícil definir o escopo do cenário sem antes ter uma noção de todo o processo que o criou, e isso se dá de uma forma muito simples quando utilizado os casos de uso.

Os diagramas de cenários desenvolvidos estão anexos no apêndice E, e a sua implementação serviu como base para a elicitação dos requisitos da tela de lançamentos, conforme Apêndice A.

5.4 CASOS DE USO

Para dar forma a tela de lançamentos, que inicialmente foi formulada dentro da visão geral no Apêndice A por meio de brainstorm e entrevistas, foi utilizada a técnica de cenários juntamente com casos de uso, pois conforme Stiehl (2011) os cenários devem ser utilizados para dar detalhes a uma descrição geral dos requisitos já levantados, e levando em consideração que os casos de uso dão uma noção geral da interação dos atores com o sistema, ele foi utilizado para se obter a noção geral, e logo depois ser detalhado por meio de cenários.

Para Vazquez e Siqueira (2016, p. 241) um cenário “corresponde a diferentes passos que se desdobram a partir de um evento que inicia o caso de uso e das condições que afetam como ele deve se comportar.". De forma semelhante Silva e Videira (2001), definem cenários como uma instância de um caso de uso, na qual um caso de uso pode possuir dezenas de cenários. Essas teorias se comprovaram na prática, sendo que a melhor maneira para se implementar o cenário foi antes desenvolvendo um caso de uso. Após a definição do caso de uso, a implementação dele mesmo por meio de cenários torna o processo mais completo.

Os pontos positivos da utilização dos casos de uso juntamente com cenários foram que eles deram um esboço mais generalizado para a interação, já nos cenários, esse esboço se tornou algo mais concreto e compreensível.

Para visualizar os casos de uso desenvolvidos no processo de lançamentos, vide Apêndice B.

5.5 BRAINSTORMING

O brainstorm foi utilizado para formular o item “Visão Geral do Sistema” do documento de requisitos do OrganiApp (vide Apêndice A), pois inicialmente tinha-se apenas uma ideia geral do desenvolvimento de um gestor financeiro, porém sem maiores informações, indo de encontro com a teoria de Batista e Carvalho (2003) que diz que um brainstorming

possui uma alta eficácia quando se trata de uma concepção de um sistema, devido a isso essa técnica foi a escolhida e se mostrou muito eficiente.

O *brainstorm* se deu em duas etapas, a etapa de geração onde todas as ideias foram coletadas sem nenhuma crítica, e a etapa de avaliação, onde as ideias passaram por um filtro que tornou o projeto mais sólido. Essa divisão em duas etapas se deu considerando a visão de Petry, Garcia e Souza (2014) que afirmam que a divisão do brainstorm tornam-no mais criativo, pois as ideias são coletadas na primeira fase sem nenhum tipo de crítica, que muitas vezes bloqueiam a criatividade dos envolvidos.

Nessa etapa o *brainstorm* se mostrou muito produtivo, pois deu corpo ao projeto que inicialmente era apenas uma ideia, desta forma obteve-se um apanhado geral de como o sistema deve se comportar bem como as principais características esperadas pelas partes interessadas, conforme demonstra o Apêndice A.

Após a aplicação do brainstorm, percebeu-se que a teoria de Petry, Garcia e Souza (2014) se mostrou eficiente na prática, pois os stakeholders se sentiram mais à vontade para demonstrar o seu ponto de vista, e todas as ideias, por mais que não fossem utilizadas, serviram de base para criação de novas ideias. Essa análise resultou no Quadro 7.

Toda documentação levantada pelas reuniões de brainstorm estão anexos no Apêndice D.

5.6 APLICAÇÃO DO MÉTODO INDUTIVO E DEDUTIVO

Esta sessão busca demonstrar por meio dos métodos dedutivo e indutivo, com base na literatura, aplicação prática (Seção 5) e no resultado da análise da Seção 5, que é o Quadro 2, qual técnica é mais indicada para cada função no processo de elicitação de requisitos de um sistema, bem como demonstrar como fazer o melhor uso de cada ferramenta.

5.6.1 Entrevistas

Ao utilizar as entrevistas pode-se perceber como vantagem uma maior interação dos usuários e a equipe de desenvolvimento por quebrar a barreira entre stakeholders e desenvolvedores, então pode-se afirmar que ela aproxima todos os interessados. Outro ponto forte do uso de entrevista é que abrangem uma grande quantidade de interessados de uma vez só, desta forma facilitam a compreensão geral do sistema e de como os usuários/interessados pretendem interagir com o sistema. Além disto entrevistas conseguem extrair com mais eficiência informações subjetivas a respeito do sistema.

Para um maior aproveitamento das técnicas e benefícios da entrevista é importante aplicar metodologias de uso e estruturação das mesmas, a estruturação da entrevista além de quebrar o gelo consegue focar o conteúdo da entrevista tirando um maior aproveitamento.

Em contrapartida é importante frisar que quando muitos interessados são entrevistados o custo/tempo investido podem acabar prejudicando o projeto. Em anexo a isto os usuários especialistas podem acabar não expondo determinadas informações cruciais para o software por ter aquilo como rotina e julgar tais conhecimentos implícitos, além disso as terminologias utilizadas podem dificultar o trabalho dos engenheiros de requisitos.

A utilização de entrevistas facilita a junção de informação útil com diferentes pontos de vista, além de facilitar o entendimento de como os usuários imaginam o sistema e suas interações.

Percebeu-se com a aplicação da entrevista no projeto que ela não é um processo completo, então pode-se afirmar que a mesma deve ser mesclada com outras técnicas de elicitação de requisitos para que assim venha a se tirar o máximo de proveito. Essa técnica pode ser utilizada em todas as etapas da elicitação de requisitos, porém deve ser utilizada juntamente com outra técnica, que deve ser escolhida tendo como base a etapa da elicitação de requisitos.

Quadro 2 – Vantagens e desvantagens da técnica entrevista.

Técnica	Entrevista	
	Vantagens	**Desvantagens**
Teoria	1) Aproxima usuários e desenvolvedores (MARTINS, 2001). 2) Abrangem grande quantidade de interessados com diferentes pontos de vista (KENDALL; KENDALL, 1992). 3) Extraem com mais eficiência informações de caráter subjetivo (MARTINS, 2001).	1) Custo e tempo investidos podem ser prejudiciais ao projeto (MARTINS, 2001). 2) Usuários podem não expor determinados conhecimentos por estarem habituados a ele (SOMMERVILLE, 2011). 3) As terminologias utilizadas pelos usuários podem

	4) Facilitam a compreensão geral do sistema (SOMMERVILLE, 2011). 5) Auxilia na compreensão de como os usuários pretendem interagir com o sistema (SOMMERVILLE, 2011).	dificultar o entendimento do engenheiro de requisitos (SOMMERVILLE, 2011).
Prática	Facilita a junção de informação útil com diferentes pontos de vista, conforme Kendall e Kendall (1992), além de facilitar o entendimento de como os usuários imaginam o sistema e suas interações (SOMMERVILLE, 2011).	Durante as entrevistas, os entrevistados ficam receosos de falar, sendo difícil contornar esta situação pois apenas estruturar a entrevista conforme Vazquez e Siqueira (2016) não é o suficiente, o entrevistador precisa ter desenvoltura para conseguir "quebrar o gelo". Outro fator que dificulta esta técnica é definir uma linguagem comum entre entrevistados e entrevistador.

Fonte: autores.

5.6.2 Etnografia

A técnica de etnografia é excelente para ajudar a compreender a real necessidade do cliente bem como suas dificuldades e o processo realizado, onde ficou muito mais simples de elicitar requisitos implícitos ao processo. Isso se deve ao fato desta técnica focar no usuário final, permitindo assim que detalhes cruciais do processo sejam capturados e documentados.

Por outro lado, é uma técnica limitada pois necessita de algo a ser observado, um processo existente, além disso é um processo complexo e que demanda de muito tempo podendo levar ao fracasso devido ao investimento. Em complemento a isto é importante frisar que não é uma técnica adequada para elicitar requisitos de domínio da aplicação e não é eficaz para documentar possíveis melhorias pois o seu foco é estritamente no usuário final e ainda alguns processos que ocorrem esporadicamente podem não ser observados nas janelas de observação.

Para um melhor aproveitamento desta técnica é necessário entender que sua real finalidade é o levantamento de requisitos implícitos e compreender o contexto social e organizacional do software que está sendo desenvolvido.

Observar os usuários na realização de suas atividades permite visualizar e documentar requisitos implícitos conforme previsto, e visualizar passos comuns aos usuários que não eram explicitados durante entrevistas por exemplo, devido a isso, essa técnica não deve ser utilizada de forma isolada, e sua utilização deve ocorrer para elicitar processos específicos do software.

Outro ponto a ser levado em consideração é que o tempo investido é alto, e ainda deve-se considerar que os dados devem ser analisados e documentados, de documentação e análise dos dados obtidos o qual torna o processo muito oneroso

Quadro 3 – Vantagens e desvantagens da técnica etnografia.

Técnica	Etnografia	
	Vantagens	**Desvantagens**
Teoria	1) Ajuda no levantamento de requisitos implícitos os quais refletem processos reais (FABRI, 2012) 2) Facilita o entendimento da real necessidade do cliente, bem como as dificuldades enfrentadas, a importância e criticidade	1) Limitado a processos existentes (VAZQUEZ; SIQUEIRA, 2016). 2) Processo é muito oneroso demandando muito tempo (VAZQUEZ SIQUEIRA, 2016). 3) Processos que ocorrem com baixa frequência

	do trabalho a ser realizado (VAZQUEZ; SIQUEIRA, 2016). 3) Foco no usuário final permite compreender detalhes cruciais de processos (SOMMERVILLE, 2011).	podem não ser observados (VAZQUEZ; SIQUEIRA, 2016). 4) Não adequada para levantar requisitos de domínio da aplicação (SOMMERVILLE, 2011).
Prática	Observar os usuários na realização de suas atividades permitiu visualizar e documentar requisitos implícitos conforme previsto por Fabri (2012) e Sommerville (2011), passos comuns aos usuários que não eram explicitados durante entrevistas por exemplo.	O tempo investido na observação é muito alto, fora o processo de documentação e análise dos dados obtidos o qual torna o processo muito oneroso, conforme previsto por Vazquez e Siqueira (2016).

Fonte: autores.

5.6.3 Cenários

Durante a aplicação dos diagramas de cenários foi possível perceber que fica difícil definir um cenário sem antes ter uma noção geral do sistema, devido a isso, essa técnica deve ser utilizada em fases finais da elucidação de requisitos, pois anteriormente a sua aplicação é necessário utilizar de técnicas como o brainstorm para formar um apanhado geral do funcionamento do sistema.

Os casos de uso são a melhor forma de definir escopos aos cenários, isso foi percebido pois o cenário é uma interação específica entre os atores, e os casos de uso definem de forma mais geral as interações entre os atores e o sistema.

Os cenários definem de forma detalhada as interações do usuário, dessa forma, pode-se afirmar que ele auxilia no entendimento de todos os stakeholders ao sistema, aumentando a qualidade do mesmo e diminuindo custos de retrabalho futuramente.

Quadro 4 – Vantagens e desvantagens da técnica cenários.

Técnica	Cenários	
	Vantagens	**Desvantagens**
Teoria	1) Detalham com riqueza descrições gerais já elicitados (STIEHL, 2011). 2) Grande interação com os stakeholders (VAZQUEZ; SIQUEIRA, 2016). 3) Identifica possíveis roteiros de uso (SOMMERVILLE, 2011). 4) Identifica diferentes fluxos dos eventos, prevendo falhas e processos simultâneos (SOMMERVILLE, 2011).	1) Necessita de uma descrição geral dos requisitos (STIEHL, 2011).
Prática	A aplicação de cenários auxilia muito no processo de elucidação, mas também fora disso, auxilia no entendimento do que precisa ser realmente desenvolvido, trazendo uma riqueza de detalhes de crucial importância para a qualidade do sistema, algo previsto por Stiehl (2011). Além disso possui uma escrita simplificada em relação aos casos de uso mantendo uma eficiência equivalente, além disso consegue descrever de	É necessário ter um processo funcional para que as informações sejam analisadas (SOMMERVILLE, 2011). Outra dificuldade encontrada foi a de não prever a maneira de como as informações devem ser obtidas, dificultando o trabalho do analista de requisitos.

	maneira mais simples diferentes fluxos do processo conforme exposto por Sommerville (2011).	

Fonte: autores.

5.6.4 Casos de Uso

Durante a aplicação dos casos de uso, observou-se que a compreensão do sistema por parte dos stakeholders se deu com maior facilidade, então pode-se afirmar que essa técnica deve ser utilizada para que todos conheçam de uma forma mais sistema os processos do sistema.

Um ponto negativo observado é que os usuários demoram um pouco para entenderem o diagrama de casos de uso, desta forma a sua execução sofre rejeição no início aplicação.

Os documentos levantados por meio dos casos de uso são muito completos, com isso pode-se afirmar que além da elicitação de requisitos eles podem ser utilizados para criação de manuais, na condução do desenvolvimento e servir inclusivamente como auxílio para eventuais suportes.

O caso de uso deveria ser aplicado em todas as etapas do desenvolvimento de um sistema devido a sua importância, porém com o custo para a implementação de tal, alguns desenvolvedores acabam ignorando essa etapa da elicitação.

Quadro 5 – Vantagens e desvantagens da técnica caso de uso.

Técnica	Caso de Uso	
	Vantagens	**Desvantagens**
Teoria	1) Ajudam na condução de todo o processo de desenvolvimento (SILVA; VIDEIRA, 2001). 2) Permite a compreensão total do funcionamento de uma interação bem como os	1) Não descreve a maneira como os casos de uso devem ser capturados (FOWLER, 2005). 2) Caso a documentação seja fraca, o caso de uso pode fornecer uma análise fraca das características

	seus impactos (SOMMERVILLE, 2011). 3) Facilitam a iniciação da modelagem de requisitos (SOMMERVILLE, 2011).	impactadas (SOMMERVILLE, 2011).
Prática	Aplicar esta técnica auxilia muito a compreensão de como os processos devem ocorrer dentro do sistema e as suas interações, ações em caso de erros e compreender os interessados naquele processo (SOMMERVILLE, 2011). O documento levantado por meio do caso de uso se torna útil durante todo o processo de desenvolvimento, pois pode auxiliar na criação de manuais, na condução do desenvolvimento e serve inclusive como auxílio para eventuais suportes (SILVA; VIDEIRA, 2001).	O sucesso da aplicação da técnica depende exclusivamente da documentação textual efetuada, pois se algum passo do processo não for previsto pode criar lacunas no desenvolvimento e possíveis retrabalhos (SOMMERVILLE, 2011). Também foi percebido que os usuários demoram um tempo para se habituar com os diagramas.

Fonte: autores.

5.6.5 Brainstorming

Outra técnica aplicada foi o Brainstorm a qual se mostrou extremamente útil para concepção de um projeto, pois além de estimular ideias criativas com diferentes pontos de vista, ela conseguiu extrair uma grande quantidade de informações.

Além disso possui um custo de aplicação baixo e pode ser realizado de maneira rápida com sessões que levam em média uma hora.

Percebeu-se que o brainstorm pode ser utilizado para qualquer etapa da elicitação de requisitos, porém se mostrou mais indicado para fases iniciais do projeto, pois consegue elicitar informações gerais do sistema com um grande aparato de opiniões.

Devido a necessidade de se ter vários stakeholders reunidos, essa técnica pode ser um pouco custosa.

Quadro 6 – Vantagens e desvantagens da técnica brainstorm.

Técnica	Brainstorm	
	Vantagens	**Desvantagens**
Teoria	1) Estimula a criação de ideias e soluções criativas com diferentes pontos de vista (BATISTA; CARVALHO, 2003). 2) Extrai uma grande quantidade de informações úteis para serem analisadas (BATISTA; CARVALHO, 2003). 3) Alta eficácia quando se fala em concepção do sistema, pois é nesta etapa que se está mais carente de ideias (BATISTA; CARVALHO, 2003). 4) Pode ser realizado de maneira rápida, com sessões de em média uma hora, sendo considerado um custo baixo (BATISTA; CARVALHO, 2003).	1) Tem alto custo de esforço pois necessita reunir um grupo significativo de pessoas (BATISTA; CARVALHO, 2003). 2) A reunião pode ser comprometida facilmente caso aconteça algum julgamento ou crítica da ideia apresentada (BATISTA; CARVALHO, 2003). 3) Os participantes podem não se sentir à vontade para contribuinte com tudo o que sabem (GOGUEN, 1997).

Prática	Utilizar brainstorm é ótimo para obter informações gerais do sistema, quais as possíveis funcionalidades bem como as mesmas devem funcionar e isso se deve a diversidade de opiniões "fora da caixa" que podem ser obtidas por meio das reuniões, facilitando muito a concepção do sistema conforme previsto por Batista e Carvalho (2003).	Por necessitar de várias pessoas reunidas em um mesmo tempo o custo de esforço é o ponto que mais prejudica esta técnica, situação a qual é prevista por Batista e Carvalho (2003).

Fonte: autores.

6 CONSIDERAÇÕES

O reconhecimento e a compreensão da real necessidade do cliente é uma das tarefas mais complexas desempenhadas pela engenharia de requisitos na criação de um sistema, portanto, pode-se chegar à conclusão que para garantir a qualidade de um sistema e a produção de uma documentação eficaz é extremamente necessário que o processo de elicitação de requisitos aconteça no início do projeto evitando que a descoberta dos requisitos seja efetuada de forma gradual o que pode gerar um retrabalho tanto no desenvolvimento quanto na documentação dos requisitos ou ainda criar rotinas que não estejam de acordo com as reais necessidades dos interessados.

Levando este fato em consideração este trabalho objetivou elencar as principais técnicas de elicitação de requisitos e efetuar uma análise de sua aplicação no processo de elicitação de requisitos de um sistema de gerenciamento financeiro pessoal.

A etapa de aplicação das técnicas elencadas gerou, de forma prática, uma documentação dos requisitos que é o guia do que deve ser implementado pela equipe de desenvolvimento, contudo ao aplicar as práticas obteve-se resultados práticos de como as diferentes técnicas de elucidação de requisitos se comportam perante uma aplicação prática dos conceitos estudados.

Desta forma, das técnicas estudadas conclui-se que todas são aptas para o processo de levantamento de requisitos, contudo para se obter um grau de cobertura confiável é necessário que se apliquem mais de uma técnica no levantamento de requisitos respeitando desta forma as limitações de cada uma das técnicas, com isso, foi atingido o objetivo do trabalho de analisar as técnicas de elicitação de requisitos.

Conforme Quadro 7, algumas formas de tirar melhor proveito das técnicas elencadas foram listadas de maneira a otimizar a quantidade e a qualidade das informações levantadas por meio de cada técnica a ser utilizada, levando em consideração sempre que cada sistema, interessado e organização se comportam de maneira diferente, sendo necessário então a adaptação das técnicas elencadas de modo a atender as necessidades da equipe de desenvolvimento.

Por conseguinte, acerca dos objetivos específicos deste presente trabalho considera-se:

- Elencar, com base na literatura, técnicas de engenharia de requisitos: atingido por meio da revisão bibliográfica de vários autores renomados na engenharia de software e da engenharia de requisitos;

- Aplicar as técnicas elencadas na elicitação de requisitos de um sistema de gerenciamento financeiro pessoal: atingido por meio da criação de um documento de requisitos da aplicação OrganiApp;
- Analisar as técnicas elencadas: atingido por meio da aplicação do método dedutivo efetuado após a aplicação prática das técnicas elencadas.

Quadro 7 – Quadro de recomendações de uso das técnicas.

Técnica	Recomendação de uso
Entrevista	A elicitação de requisitos por meio de entrevistas não é um processo completo, não atingindo algumas informações essenciais. Desta forma, a sua utilização juntamente com outras técnicas de elicitação de requisitos deve ser um fator a levar em consideração, técnicas essas as quais a equipe de engenharia de requisitos julgar adequadas ao sistema que está sendo desenvolvido (SOMMERVILLE, 2011).
Etnografia	A técnica de etnografia não é adequada para elicitação de requisitos de domínio, e muita das vezes a inserção no ambiente de trabalho não consegue identificar novos recursos que podem ser inseridos no sistema, dessa forma, ela se torna uma técnica incompleta. Com base nisso, assim como nas entrevistas ela deve ser mesclada com outras técnicas de elicitação, para que assim venha-se a extrair o máximo de aproveitamento desta ferramenta (SOMMERVILLE, 2011).
Brainstorm	Para ser melhor aproveitada, a técnica de brainstorming deve ser dividida em duas etapas. Na primeira etapa, chamada de Geração de Ideias, os envolvidos são inibidos de realizar qualquer tipo de crítica ou avaliação, pois mesmo uma ideia não sendo boa na visão de alguns, ela pode servir de base para criação de novas ideias. Na segunda etapa, chamada de Avaliação, as ideias coletadas são discutidas entre os *stakeholders*. Essa separação torna o processo mais criativo, mas

	mesmo assim com um filtro, para que as ideias desnecessárias sejam descartadas ao fim. (PETRY; GARCIA; SOUZA, 2014).
Caso de Uso	De modo a obter uma análise completa de todas as características que serão impactadas por um requisito é necessário que todos os casos de uso levantados sigam como boa prática possuir uma documentação textual completa e detalhada. Desta forma o entendimento das interações e impactos de uma alteração são mais completos, podendo ainda anexar outros documentos ou visões de forma a complementar a documentação (SOMMERVILLE, 2011).
Cenário	Para Vazquez e Siqueira (2016, p. 241) um cenário "corresponde a diferentes passos que se desdobram a partir de um evento que inicia o caso de uso e das condições que afetam como ele deve se comportar.". De forma semelhante Silva e Videira (2001), definem cenários como uma instância de um caso de uso, na qual um caso de uso pode possuir dezenas de cenários. Essas teorias se comprovaram na prática, sendo que a melhor maneira para se implementar o cenário foi antes desenvolvendo um caso de uso. Após a implementação do caso de uso, a implementação do mesmo por meio de cenários torna o processo mais completo. Fica difícil definir o escopo do cenário sem antes montar todo o processo que o criou, e isso se dá de uma forma muito simples quando utilizado os casos de uso.

Fonte: autores.

REFERÊNCIAS

AMBLER, Scott W. **Agile Documentation**. 2001-2004, The Official Agile Modeling (AM) Site, 2001, Disponível em: <http://www.agilemodeling.com/shared/AMOverview.pdf>, Acesso em: 02 set. 2018.

BALTHAZAR, Glauber da Rocha (Ed.). Visão Geral da Qualidade de Software. **Revista Eletrônica da Faculdade Metodista Granbery**, Juiz de Fora, p.1-11, 3 dez. 2007. Disponível em: <http://re.granbery.edu.br/artigos/MjUw.pdf>. Acesso em: 30 out. 2018.

BATISTA, Edinelson A.; CARVALHO, Ariadne M. B. R. **Uma Taxonomia Facetada para Técnicas de Elicitação de Requisitos.** Encontrado em http://www.inf.pucrio.br/wer/WERpapers/artigos/artigos_WER03/edinelson_batista .pdf. Acesso em 01/02/2018

BRYMAN, Alan. **Social research methods**. Oxford university press, 2016.

BUENO, Cassiane de Fátima dos Santos; CAMPELO, Gustavo Bueno. Qualidade de Software. **Departamento de Informática**, Recife, p.1-28, 1999.

CARVALHO, Ana Elizabete Souza de; TAVARES, Helena Cristina; CASTRO, Jaelson Brelaz. **Uma Estratégia para Implantação de uma Gerência de Requisitos Visando a Melhoria Dos Processos de Software.** 2001. 23 f. TCC (Graduação) - Curso de Gestão em Redes de Computadores, Centro de Informática, Universidade Federal de Pernambuco, Recife, 2001.

COSTA, La; ZOUCAS, Alessandra. **Elicitação de Requisitos de Software no Setor Público:** Lições Aprendidas e Recomendações para Mitigação de Riscos. Simpósio de Excelência em Gestão de Tecnologia, Resende, p.1-14, 2012. Disponível em: <https://www.aedb.br/seget/arquivos/artigos12/42116413.pdf>. Acesso em: 30 out. 2018.

COUTINHO, Ítalo. **Comunicando-se com eficácia com os stakeholders.** PMKB - Project Management Knowledge Base, Bom

Despacho, p.1-1, 2012. Disponível em: <https://pmkb.com.br/artigos/comunicando-se-com-eficacia-com-os-stakeholders/>. Acesso em: 30 out. 2018. Dissertação (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e Computação, Universidade Estadual de Campinas.

CRESWELL, John W. **Projeto de Pesquisa: Métodos qualitativo, quantitativo e misto.** 2. ed. Porto Alegre: Artmed Editora S.a., 2007.

ENGHOLM JÚNIOR, H. **Engenharia de Software na prática.** São Paulo: Novatec, 2010.

FABRI, José Augusto. **A etnografia como processo de levantamento de requisitos.** 2012. Disponível em: <https://engenhariasoftware.wordpress.com/2012/12/11/a-etnografia-como-processo-de-levantamento-de-requisitos/>. Acesso em: 30 out. 2018.

FACHIN, Odília. **Fundamentos de Metodologia.** 5. ed. São Paulo: Saraiva, 2005.

FERENHOF, Hélio Aisenberg; FERNANDES, Roberto Fabiano. **Desmistificando a revisão de literatura como base para redação científica: Método SSF.** 2016. 8 f. Monografia (Especialização) - Curso de Engenharia de Produtos, Processos e Serviços, Universidade Federal de Santa Catarina, Florianopolis, 2016

FOWLER, M. **UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos.** 3. ed. Porto Alegre: Bookman, 2005

GOGUEN, J. A. **Techniques for requirements elicitation. In: Software Requirements Engineering**. IEEE-Computer Society Press, 2. ed. pp. 110-122, 1997.

HULL, E.; JACKSON, K.; DICK, J. **Requirements Engineering.** 3. ed. Londres: Springer, 2011.

KENDALL, K.E.; KENDALL, J.E. **Systems Analysis and Design**, Prentice Hall: 1992.

MARTINS, Luiz E. G. **Uma Metodologia de Elicitação de Requisitos de Software Baseada na Teoria da Atividade.** Campinas – SP, 2001.

MORAES, Edson Andrade de. **Utilização de Uma Estratégia Para Identificação de Fontes de Informação na Fase de Elicitação.** 2010. 1 v. Tese (Doutorado) - Curso de Programa de Pós-Graduação em Informática, PontifÍcia Universidade CatÓlica do Rio de Janeiro - Puc-rio, Rio de Janeiro, 2010. Disponível em: <https://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=15760@1>. Acesso em: 30 out. 2018.

PETRY, Kleber Lopes; GARCIA, Éder Moretto; SOUZA, Rodrigo Clemente Thom de. Levantamento de Requisitos de Forma Enxuta. **II Seminário Empresarial e II Jornada de Ti da Faculdade Cidade Verde.** Maringá, p. 1-6. out. 2014. Disponível em: <http://fcv.edu.br/admin/assets/repositorio_arquivo/6a8143e2f444e901e3f9700345f78e90.pdf>. Acesso em: 02 nov. 2018.

PRESSMAN, R. S. **Engenharia de Software: uma abordagem profissional.** 7. ed. Porto Alegre: McGraw-Hill, 2011.
PRESSMAN, Roger S**. Engenharia de Software.** São Paulo: Makron Books, 1995.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico.** 2. ed. Novo Hamburgo: Editora Feevale, 2013. 277p.

ROZANSKI, Nick. **Software Systems Architecture:** Working with Stakeholders Using Viewpoints and Perspectives. 2. ed. Boston: Addison-wesley, 2012.

SHAMIEH, Cathleen. **Engenharia de Requisitos para Leigos.** Hoboken: Wiley Publishing, 2012.

SILVA, Alberto; VIDEIRA, Carlos. **UML CASE**: Metodologias e Ferramentas. Portugal: Centro Atlântico PT, 2001. 578 p.

SOMMERVILLE, I. **Engenharia de Software.** 9. ed. São Paulo: Pearson Prentice Hall, 2011.

STIEHL, Diego. **Aplicação da Engenharia de Requisitos no Estudo Experimental de um sistema de Gerência de Eventos.** 2011. 144 f. Monografia (Especialização) - Curso de Pós-Graduação em Engenharia de Software, Universidade Tecnológica Federal do Paraná, Medianeira, 2011. Cap. 2 Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/1117/3/MD_ENGESS_I_2012_08.pdf>. Acesso em: 09 set. 2018.

TAVARES, Ana Lúcia de Oliveira. **Engenharia de Software: uma visão geral.** 2006. 13 f. Monografia (Especialização) - Curso de Curso de Especialização em Engenharia de Projetos de Software, Unisul, Palhoça, 2006. Disponível em: <http://www.joinville.udesc.br/portal/professores/claudinei/materiais/SOFT_VISAO_GERAL.pdf>. Acesso em: 30 out. 2018.

VAZQUEZ, Carlos Eduardo; SIQUEIRA, Guilherme Simões. **Engenharia de Requisitos:** software orientado ao negócio. São Paulo: Brasport, 2016. 328 p.

ZAVE, P. **Classification of Research Efforts in Requirements Engineering.** ACM Computing Surveys, 29, Dezembro 1997. 315-321.

ZOPPA, Alexandre. **Stakeholders: por que eles são importantes?** PMKB - Project Management Knowledge Base, Bom Despacho, p.1-1, 14 out. 2013. Disponível em: <https://pmkb.com.br/artigos/stakeholders-por-que-eles-sao-importantes/>. Acesso em: 30 out. 2018.

APÊNDICE A – Documento de Requisitos

DOCUMENTO DE REQUISITOS DO SOFTWARE ORGANIAPP - GESTÃO FINANCEIRA PESSOAL

Afim de verificar a aplicabilidade prática das técnicas de especificação de requisitos, este documento busca especificar os requisitos necessários para o desenvolvimento de um aplicativo/*software* denominado OrganiApp - Gestão Financeira Pessoal. Para tal é apresentada uma visão geral do sistema, com o seu propósito e principais características e na sequência são explicitados os requisitos funcionais e não funcionais.

Prioridades de requisitos

Para estabelecimento da prioridade dos requisitos foram adotadas 3 denominações sendo elas:

- **Essencial:** requisitos imprescindíveis para o funcionamento do sistema desde a primeira versão.
- **Importante:** requisitos sem os quais o sistema funciona, porém não de maneira satisfatória.
- **Desejável:** requisitos que não influenciam o funcionamento do sistema, podendo ser implementados por último.

VISÃO GERAL DO SISTEMA

O OrganiApp - Gestão Financeira Pessoal é um sistema de gestão de recursos financeiros pessoal que visa utilizar das mais recentes tecnologias web para melhorar a gestão pessoal de recursos, tornando o acesso e lançamento de gastos e recebimentos rápido, intuitivo e eficiente.

O sistema deverá contar com três macro ambientes sendo eles a visão geral do sistema, os lançamentos e os relatórios. Cada macro ambiente logicamente relacionado com os demais.

O OrganiApp prevê uma visão geral para o usuário logado com gráficos e informações pertinentes como o saldo geral do usuário, os cartões vinculados e botões de acesso rápido a rotina de lançamentos bem como suas sub-rotinas como lançamentos de débitos ou créditos a rotina de relatórios bem como as configurações do aplicativo.

Na seção de lançamentos o OrganiApp deve disponibilizar de rotinas para o lançamento de débitos e créditos bem como uma listagem de maneira a expor apenas informações relevantes sobre os lançamentos do mês.

Na aba de relatórios o sistema deverá contar com filtros intuitivos para geração de relatórios de maneira a flexibilizar a gestão e controle financeiro do usuário.

Ademais desses três macro ambientes, o sistema deverá prever uma página de configuração onde o usuário poderá definir quais notificações deseja receber, criar e gerenciar categorias de lançamentos, criar e gerenciar 'contas' pessoais e demais configurações relativas ao aplicativo.

O OrganiApp deverá utilizar de tecnologias de ponta de forma a garantir a interoperabilidade entre diferentes sistemas operacionais, utilizando da computação em nuvem para que haja homogeneidade dos dados, além de contar com um sistema stand alone que sincronizará todos os lançamentos feitos offline, de forma que, ao restabelecer a conexão com a internet os dados sejam integrados ao sistema.

CARACTERÍSTICAS

Dashboard

O Dashboard pode ser considerado a parte mais importante do OrganiApp, pois por meio dele o usuário pode ter acesso a um resumo de todas as informações que ele tem interesse, sem que seja necessário a navegação sob muitos itens do sistema.

Este módulo visa agregar ao usuário informações cruciais para uma boa gestão de seus recursos financeiros, deverá contar com gráficos informativos, acesso a cadastro de categorias, acesso rápido para lançamento de despesas, receitas, saldo atual e previsto para o período e outras e configurações pertinentes para uma gestão eficiente dos recursos do usuário.

Lançamentos

O módulo de lançamentos deverá conter todas as rotinas inerentes a manutenção do livro caixa do usuário sendo elas lançamentos de despesas e receitas.

Durante o lançamento de movimentações o sistema deverá prever algumas configurações para o lançamento conforme abaixo:

Da forma de pagamento

- **À vista:** O lançamento irá levar em consideração a data informada na rotina.
- **Parcelado:** O sistema deverá solicitar a quantidade de parcelas e a data da cobrança das mesmas.
- **Fixo:** Lançamentos que são pagos mensalmente, mas não possuem um limite definido de parcelas, como por exemplo o seguro de um automóvel.

Da categorização

O sistema deverá permitir criar e gerenciar categorias customizáveis de lançamentos de maneira a permitir ao usuário um maior controle sobre seus gastos e receitas. De forma genérica, quando nenhuma categoria for criada ou selecionada na hora do lançamento, o sistema assumirá o lançamento como sendo da categoria padrão "Outros".

Da data

A aplicação deverá permitir ao usuário informar qual a data do lançamento, permitindo lançamentos retroativos. Quando de lançamentos parcelados o sistema automaticamente lançará nos meses subsequentes parcelas com a mesma data de vencimento.

Do valor

O valor informado no lançamento se refere ao valor a ser cobrado naquele mês, sendo assim em compras parceladas deverá ser informado o valor da parcela e não o total da compra.

Relatórios

A partir da análise das informações obtidas com a etnografia, definiu-se que o módulo de relatórios deverá disponibilizar de filtragens que tornem flexível a exibição dos lançamentos de maneira a atender as necessidades do usuário.

A rotina deverá contar com filtros que possibilitem ao usuário organizar por data, tipo de lançamento e categorias os lançamentos do mesmo.

OBJETIVOS DO PROJETO

O projeto OrganiApp tem por objetivo geral a implementação de um aplicativo multiplataforma integrado via nuvem para gestão financeira pessoal, visando assim, trazer ao usuário uma organização financeira mais assertiva.

Os objetivos específicos são:

1. Auxiliar no controle das finanças pessoais do cliente;
2. Integrar via nuvem os dados do usuário que estejam em diferentes sistemas operacionais;
3. Facilitar no lançamento de despesas e receitas;
4. Disponibilizar relatórios sintéticos com informações cruciais.
5. Criar serviço standalone que sincronize os dados lançados offline ao primeiro contato com a internet.

REQUISITOS FUNCIONAIS

Requisitos funcionais do sistema OrganiApp elencados por meio das técnicas de elicitação de requisitos.

[RF001] Fazer Logon
O ambiente disponibiliza uma operação que autentica um usuário e verifica as funcionalidades do ambiente que estão disponíveis ao mesmo de acordo com a política de autenticação e autorização.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RF002] Fazer Logoff
O ambiente disponibiliza uma funcionalidade de encerrar a sessão do usuário com o ambiente.
Prioridade: **( )** Essencial (X) Importante ( ) Desejável

[RF003] Cadastrar Despesas
O ambiente disponibiliza uma operação para cadastrar despesas que podem ser do tipo parcelada, fixa ou apenas com desconto no mês vigente. Essa despesa tem valor, categoria, descrição e data.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RF004] Cadastrar Receitas
O ambiente disponibiliza uma operação para cadastrar créditos que podem ser do tipo parcelado, fixo ou apenas com crédito no mês vigente. O crédito possui data, categoria, valor e descrição.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RF005] Emitir relatório mensal
O ambiente disponibiliza uma operação que permite a visualização dos dados cadastrados no mês para que o usuário possa acompanhar uma espécie de extrato mensal.
Prioridade: **( )** Essencial (X) Importante ( ) Desejável

[RF006] Tela de visão geral
O ambiente disponibiliza uma tela de visão geral com um resumo dos lançamentos mensais de forma que o usuário não necessite emitir nenhum relatório para visualizar dados essenciais do sistema.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RF007] Editar dados pessoais
O ambiente permite que usuários cadastrados possam modificar seus dados pessoais.
Prioridade: **( )** Essencial ( ) Importante (X) Desejável

[RF008] Excluir conta
O ambiente permite que os usuários excluam suas contas no aplicativo, sendo que todos os seus dados e históricos são apagados.
Prioridade: **( )** Essencial ( ) Importante (X) Desejável

[RF009] Lembretes de despesas
O ambiente disponibiliza o cadastramento de data de pagamento nas despesas, para que o mesmo seja alertado que deve pagá-la. Esse alerta deve ser em forma de e-mail e notificação no aplicativo.
Prioridade: **( )** Essencial ( ) Importante (X) Desejável

[RF010] Sistema de metas (despesas)
O ambiente disponibiliza um mecanismo que possibilita um usuário definir uma meta de gastos totais em determinada categoria de despesa.
Prioridade: **( )** Essencial (X) Importante ( ) Desejável

[RF011] Sistema de metas (receitas)
O ambiente disponibiliza um mecanismo que possibilita um usuário definir uma meta de ganhos totais em determinada categoria de crédito.
Prioridade: **( )** Essencial ( ) Importante (X) Desejável

[RF012] Lembrar senha
O ambiente disponibiliza um mecanismo que possibilita um usuário lembrar sua senha.
Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF013]Alterar senha
O ambiente disponibiliza um mecanismo que possibilita um usuário alterar sua senha.
Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF014]Relatório gráfico de categoria de despesas
O ambiente disponibiliza um relatório contendo um gráfico onde demonstra os percentuais de cada categoria de despesa em um determinado período para que o usuário visualize as categorias onde estão suas maiores/menores despesas mensais.
Prioridade: **( )** Essencial (X) Importante ( ) Desejável

[RF015]Relatório gráfico de categoria de receitas
O ambiente disponibiliza um relatório contendo um gráfico onde demonstra os percentuais de cada categoria de receita em um determinado período para que o usuário visualize as categorias onde estão seus maiores/menores receitas mensais.
Prioridade: **( )** Essencial (X) Importante ( ) Desejável

REQUISITOS NÃO FUNCIONAIS

Requisitos não funcionais do sistema OrganiApp elencados por meio das técnicas de elicitação de requisitos.

Flexibilidade

Com o intuito de flexibilizar a aplicação, os requisitos não funcionais citados abaixo devem ser cumpridos.

[RNF001] Política de criação de contas
Para aceder ao sistema, obrigatoriamente, duas informações devem ser fornecidas: o email e uma senha.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RNF002] Tecnologias utilizadas
O Sistema deverá ser desenvolvido em uma plataforma CLOUD, de maneira que se torne multiplataforma, podendo ser acessada de computadores, Android e IOS.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

Confiabilidade

Com intuito de manter a confiabilidade do sistema, a plataforma deve funcionar como um serviço 24x7, desta forma a confiabilidade deve ser tratada como prioritária, tendo isto em vista os requisitos não funcionais deste tópico devem ser atendidos.

[RNF003] Disponibilidade
O sistema deve estar sempre disponível ao usuário todos os dias da semana, 24 horas por dia. Quando houver necessidade de paradas estratégicas para manutenções de rotina as mesmas deverão ser programadas com antecedência e os usuários deverão ser informados.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RNF004] Recuperação de falhas
Em caso de falhas, o processo de recuperação do ambiente deve ser realizado de forma automática.
Prioridade: **( )** Essencial (**X**) Importante ( ) Desejável

Segurança

Como princípio básico de funcionamento, o ambiente deverá manter as informações de cada usuário de maneira a manter o sigilo das informações bem como a segurança contra ataques. Desta forma o sistema deverá cumprir os requisitos não funcionais abaixo.

[RNF005] Política de criação de senhas
Deverá haver regras para o procedimento de criação de senhas do usuário.
Prioridade: **( )** Essencial **(X)** Importante **( )** Desejável

[RNF006] Obrigatoriedade do Logon
Todo usuário cadastrado deverá fazer logon ao menos uma para aceder ao sistema, abrindo a possibilidade para manter o mesmo logado naquele dispositivo.
Prioridade: **( )** Essencial **(X)** Importante **( )** Desejável

[RNF007] Política de criptografia dos dados
Dados sensíveis como números de cartão e senha do usuário devem ser criptografadas.
Prioridade: **(X)** Essencial **( )** Importante **( )** Desejável

[RNF008] Finalizar sessão
Salvo em casos que o dispositivo esteja autorizado a manter-se logado, ao fechar a aplicação a sessão deverá ser finalizada.
Prioridade: **( )** Essencial **( )** Importante **(X)** Desejável

[RNF009] Sigilo dos dados
O sistema deverá prezar pelo sigilo das informações de modo que os dados de um usuário não possam ser visualizados por outros usuários.
Prioridade: **(X)** Essencial **( )** Importante **( )** Desejável

[RNF010] Integridade
O sistema deve preservar a integridade das informações cadastradas, de maneira que apenas o usuário proprietário das informações possa alterá-las mediante logon na aplicação.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RNF011] Disponibilidade
O sistema deverá garantir a disponibilidade das informações cadastradas nos sistemas, permitindo de acordo com as regras de negócio efetuar consultas e alterações nas mesmas.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

Usabilidade

Visando garantir uma boa experiência ao usuário os requisitos não funcionais descritos nesta subseção deverão ser atendidos.

[RNF012] Mensagens de erro tratadas
Todas as mensagens de erros disparadas devem ser tratadas de maneira que o usuário possa tomar as medidas necessárias
Prioridade: **( )** Essencial ( ) Importante (X) Desejável

[RNF013] Interface de fácil utilização
A interface do sistema deverá ser de fácil utilização e altamente intuitiva, de maneira que usuários leigos consigam utilizar com totalidade a ferramenta.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

[RNF014] Campos autoexplicativos
Os campos a serem informados pelos usuários devem ser intuitivos e autoexplicativos.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

Desempenho

Visando garantir o foco no usuário e no desempenho os requisitos não funcionais abaixo devem ser cumpridos.

[RNF015] Espaço em disco
O sistema deve ser implementado em um ambiente que ofereça espaço em disco suficiente para armazenar as informações cadastradas, em caso de necessidade de aumento da capacidade o sistema deverá avisar com antecedência a equipe responsável.
Prioridade: **(X)** Essencial ( ) Importante ( ) Desejável

APÊNDICE B – Diagramas de Caso de Uso

CDU001	**Efetuar Lançamento**
Ator principal	Usuário
Interesses do ator principal	Deseja efetuar o lançamento de receitas/despesas no sistema para controlar seu fluxo financeiro.
Pré-Condições	O usuário deve estar autenticado no sistema.
Pós-Condições	Os dados do lançamento devem ser armazenados no sistema.
Cenário de Sucesso Principal	1) O usuário loga no sistema. 2) O usuário seleciona a rotina de lançamentos. 3) O usuário informa o valor do lançamento. 4) O usuário informa a data a ser efetivado o lançamento. 5) O usuário informa a forma de pagamento. 6) Caso o pagamento seja do tipo parcelado o sistema abre um novo campo referente ao número de parcelas e o usuário informa o número de parcelas. O usuário seleciona a categoria do lançamento. 7) O usuário seleciona a situação do lançamento. 8) O usuário salva o lançamento.

	9) O sistema retorna uma mensagem de sucesso.
Fluxos alternativos	No ponto 1: O usuário informa login/senhas incorretas: Deve exibir mensagem de erro. No ponto 8: O sistema retorna mensagem de erro: Deve cancelar a operação

CDU002	**Alterar lançamento**
Ator principal	Usuário
Interesses do ator principal	Deseja alterar um lançamento de receitas/despesas no sistema para controlar de maneira correta seu fluxo financeiro.
Pré-Condições	O usuário deve estar autenticado no sistema.
Pós-Condições	Os dados do lançamento devem ser alterados e armazenados no sistema.
Cenário de Sucesso Principal	1) O usuário loga no sistema. 2) O usuário seleciona a rotina de lançamentos. 3) O usuário seleciona o lançamento a ser alterado. 4) O usuário seleciona a opção "Editar Lançamento". 5) O usuário altera as

	informações necessárias. 6) O usuário salva as alterações. 7) O sistema solicita confirmação de alteração. 8) O usuário confirma a alteração. 9) O sistema grava as alterações realizadas.
Fluxos alternativos	No ponto 1: O usuário informa login/senhas incorretas: Deve exibir mensagem de erro. 5 a 8: O usuário desiste da alteração: Cancelar a operação. 9: O sistema retorna mensagem de erro: Cancelar a operação.

APÊNDICE C – Entrevistas

Questionários abertos:

01 - O que você considera como informações cruciais no momento de registrar um lançamento?

02 - Em relação a usabilidade, como você imagina um sistema de controle financeiro pessoal?

03 - Como você gostaria de visualizar as informações de seus lançamentos?

04 - Que tipos de notificação você julga importante para uma gestão mais eficiente de suas finanças?

Questionários fechados:

01 - Categorizar os gastos é útil para você?
Respostas: 1. Sim 2. Não

02 - Qual das informações é mais relevante para ser exibida na visão geral do sistema?
Respostas: 1. Saldo atual 2. Saldo previsto 3. Despesas atrasadas 4. Receitas a receber

03 - Qual a plataforma você utilizaria com mais frequência em um sistema de gestão financeira pessoal?
Respostas: 1. PC (Navegadores) 2. Android 3. IOS

04 - O quão importante você considera informativos de lançamentos pendentes?
Respostas: 1. Irrelevante 2. Pouco importante 3. Importante 4. Muito importante 5. Crucial

05 - Você gostaria de receber dicas de gestão no dashboard do aplicativo?
Respostas: 1. Sim 2. Não

06 - O quão relevante você julga a disponibilização de um gráfico informativo de receitas x despesas?

Respostas: 1. Irrelevante 2. Pouco relevante 3. Relevante 4. Muito relevante 5. Crucial

APÊNDICE D – Brainstorm

Pergunta estimulada: Como você imagina um sistema de gestão financeira pessoal?

"Poder cadastrar minhas contas bancárias ou dividir minhas despesas/receitas em categorias e subcategorias traria um controle muito mais flexibilizado de minhas finanças"

"Uma interface simples, sem muitos botões, apenas o essencial para o controle financeiro"

"O sistema poderia facilitar o acesso a algumas rotinas como o cadastro de novos lançamentos."

"Trazer a informação do saldo atual seria bem útil."

"Um sistema de notificação do que está pendente de pagamento/recebimento ajudaria muito no controle das contas."

"A ideia de juntar o saldo com mais algum informativo sobre o resultado esperado do mês."

"Uma sessão de relatórios que eu possa filtrar pela data que eu preciso juntamente com categorizações dos lançamentos."

APÊNDICE E – Diagramas de Cenários

Cenário	Descrição
Autenticação Incorreta	1) Usuário abre o sistema e tenta autenticar. 2) Informa senha ou login incorretos. 3) O sistema retorna uma mensagem de erro informando que senha ou login estão incorretos. 4) Sistema solicita novamente a autenticação.
Autenticação Correta	1) Usuário abre o sistema. 2) Usuário informa login e senha corretos. 3) Sistema redireciona o usuário para a visão geral.
Efetuar Lançamento	1) O usuário autentica na rotina. 2) O usuário seleciona a opção para efetuar um novo lançamento. 3) O usuário informa o valor, a forma de pagamento, a situação e a categoria do lançamento. 4) O usuário clica em salvar. 5) O sistema deve retornar uma mensagem de sucesso.
Excluir Lançamento	1) O usuário autentica na

	rotina 2) O usuário busca o lançamento desejado entra nas opções disponíveis e seleciona excluir o lançamento. 3) O sistema solicita a confirmação da exclusão. 4) O usuário confirma a exclusão. 5) O lançamento é deletado do sistema. 6) O sistema exibe mensagem de exclusão.
Alterar situação do pagamento	1) O usuário loga no sistema. 2) O usuário busca o lançamento. 3) O usuário clica no botão da situação para alterar de pago para pendente ou vice-versa. 4) O sistema deve salvar a situação de maneira automática ao clicar no botão situação.

APÊNDICE F – Etnografia

Para aplicação da observação etnográfica foram elencados 4 usuários com vivências e maneiras diferentes de controlar seus recursos financeiros desta forma elaborou-se a tabela abaixo com as anotações efetuadas durante a janela de observação.

Usuário	Anotação
1	Para controlar suas receitas e despesas o usuário observado utiliza uma planilha a qual conta com as seguintes informações do lançamento: ● Descrição ● Valor ● Parcela ● Cartão ● Tipo (Receita/Despesa) Contendo também uma linha a qual retorna totalizadores de Receitas e Despesas bem como o saldo previsto ao final da competência (aparentemente o controle é efetuado mensalmente).
2	Utiliza de anotações em caderno para efetuar seu controle financeiro, com informações pobres anotadas consequentemente não há relatórios ou informações mais elaboradas sobre seu controle financeiro, apenas totalizadores de despesas e receitas.
3	O terceiro usuário por sua vez utiliza de um aplicativo para dispositivos móveis o qual retorna informações mais detalhadas trazendo relatórios de pizza para controle das despesas/receitas segmentando em categoria os lançamentos. Os relatórios observados ainda permitem ao usuário separar por mês as informações de lançamentos cadastradas no sistema, retornando assim um saldo previsto ao fim da competência.

4	O usuário utiliza de um aplicativo para dispositivos móveis, o qual monta relatórios de acordo com filtros preenchidos pelo usuário com filtragens por: ● Categoria ● Data ● Situação do pagamento ● Tipo do lançamento ● Formato de exibição do relatório. Em relação aos formatos possíveis de exibição o usuário pode selecionar relatórios analíticos os quais contam com informações completas dos lançamentos: ● Descrição ● Data ● Valor ● Parcela Atual ● Número de Parcelas ● Categoria ● Forma do pagamento ● Possui atrasados ● Valor total pago ● Valor total a pagar ● Valor total do lançamento Ainda permite a geração de relatórios sintéticos com informações resumidas: ● Descrição ● Valor ● Parcela (Concatenado x/y) ● Categoria ● Forma de Pagamento